Available online at www.sciencedirect.com

ScienceDirect

Computer Communications 31 (2008) 266–279

computer communications

www.elsevier.com/locate/comcom

Secure content access and replication in pure P2P networks ☆

Esther Palomar *, Juan M.E. Tapiador, Julio C. Hernandez-Castro, Arturo Ribagorda

Computer Science Department – Carlos III University Avda. Universidad 30, 28911 Leganes, Madrid, Spain

Available online 19 August 2007

Abstract

Despite the advantages offered by pure Peer-to-Peer (P2P) networks (e.g. robustness and fault tolerance), a crucial requirement is to guarantee basic security properties, such as content authenticity and integrity, as well as to enforce appropriate access control policies. These mechanisms would pave the way for new models in which content providers can exert some control over the replication and file sharing process. However, the extremely decentralized nature of these environments makes impossible to apply classic solutions that rely on some kind of fixed infrastructure, typically in the form of on-line trusted third parties (TTP). In this paper, we introduce a suite of protocols for content authentication and access control in pure P2P networks based on attribute certificates that does not rely on the existence of a public key infrastructure (PKI), privilege management infrastructure (PMI), or any other form of centralized authority. We provide an analysis concerning the efficiency (computational effort and communication overhead) and the security of our proposal.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Peer-to-Peer security; Content authentication; Public-key certificates; Access control

1. Introduction

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), ensuring that each new provider will behave accordingly to another user's access policy gets more complicated (and definitely hard should a global security infrastructure be unavailable). Particularly, it is crucial to guarantee properties such as content authenticity, as well as to enforce appropriate access control policies. In most existing P2P file sharing systems, verifying the integrity of contents depends on peers' authentication. Unfortunately, no infrastructure 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 the wrong files. This quickly makes peers lose trust and interest in the community [22]. 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.

Furthermore, many inherent characteristics of 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 [11]. For example, authorization in such environments cannot be generally provided by means of existing models, which were developed for centralized systems. 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 TTP.

The most accepted solution for authentication and authorization services in classic distributed environments relies on the existence of a PKI (or a PMI, since PKIs

☆ A preliminary version of this work appeared in the Proceedings of the 6th International Conference on P2P Computing.

* Corresponding author. Tel.: +34916248877.
E-mail addresses: epalomar@inf.uc3m.es (E. Palomar), jestevez@inf.uc3m.es (J.M.E. Tapiador), jcesar@inf.uc3m.es (J.C. Hernandez-Castro), arturo@inf.uc3m.es (A. Ribagorda).

0140-3664/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2007.08.015

are not appropriate to provide authorization services). 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.

1.1. Overview of our approach

Once a content is replicated through different locations, the originator loses control over it. Our approach allows the content owners to define and enforce restrictions on how the content is used. In a previous work [16], we introduced a protocol for content authentication in P2P networks based on public key certificates which does not rely on the existence of a PKI. The protocol relies on the scheme presented in [17], which addresses the issues related to public key authentication in P2P systems through a Byzantine agreement among nodes, not requiring for this a TTP. Our scheme maintains content integrity based on the collaboration among a fraction of peers in the system.

In this paper, we extend that work by showing how digital certificates can be used for P2P content authentication and how can be modified to provide authorization capabilities, much in the way a X.509 public-key certificate can be used as an attribute certificate. 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). According to RFC 3281 [8], 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.

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). Recent works have followed this line, studying how to use a P2P network to prevent Denial of Service (DoS) attacks on the Internet [12,15]. Furthermore, a number of approaches have been designed that support secure authentication and authorization paradigms in distributed systems, such as P2P collaborative environments [9], data-sharing P2P networks [20], and mobile ad hoc networks [5].

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

The rest of this article is organized as follows. First, Section 2 briefly overviews some related work and the background of our solution. Section 3 introduces the notation that will be used throughout the paper and a few working assumptions. In Section 4, we first present the basic operation of our scheme and subsequently discuss the three main subprotocols. Each of them will be analyzed according to security and efficiency measures in Sections 5 and 6, respectively. Finally, Section 7 concludes the paper by summarizing our contributions and outlining some future research directions.

2. Background and related work

For readability and completeness, we first overview some essential concepts used throughout this paper.

2.1. Pathak and Iftode's protocol

Pathak and Iftode [17] apply the ideas presented in the Byzantine Generals Problem [13] for providing public key authentication in pure P2P systems, where generally one cannot assume the existence of a PKI. They postulate that a correct authentication depends on an honest majority of a particular subgroup of the peers' community, labeled "trusted group". However, in this kind of systems an authenticated peer could create multiple fake identities and acts maliciously in the future (Sybil attack [6]). For this reason, the classification of the rest of the community maintained by each node has to be proactive and periodically flushed. Thus, honest members from trusted groups are used to provide a functionality similar to that of a CA (certification authority) through a consensus procedure.

The authentication protocol consists of four phases: admission request, challenge response, distributed authentication and Byzantine agreement. The protocol begins when Bob runs into a newly discovered peer, Alice, with an unauthenticated public key (K_A), and then asks for the key to a subgroup of its trusted members, in order to verify its authenticity. Each notified peer challenges Alice by sending a random nonce encrypted with Alice's supposed public key (sent by Bob) in the signed challenge message. Alice will be able to return the recovered nonce in a signed response message if and only if she holds the corresponding private key (K_A^{-1}). Each challenger waits for an application specific timeout, and if a correct response is received, he gets a proof of possession for K_A. All announced peers send their proofs of possession to Bob.

If all peers are honest, then there will be consensus and Bob will get the authentication result. Note that A or some of the peers can be detected as malicious or faulty if some votes differ. In this case, Bob first verifies if Alice is malicious by sending her the request message containing the proof. Alice must respond with all the challenge messages received and her respective responses. If Alice can prove that she is not malicious, then some of the peers must be; in that situation, Bob must communicate a Byzantine fault to the group, which will send the Byzantine agreement message to others. All these transmitted messages have timestamps, source and destination identifiers, and digital signatures. Finally, successful authentication moves a peer to the trusted group, whereas encountered malicious peers are moved to the untrusted group.

For further details, we refer the reader to the original paper in [17].

2.2. Authorization and access control models

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 data. Most collaborative systems give all participants the same rights to 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.

There are two basic types of classical access control models: discretionary and mandatory access control (DAC and MAC, respectively). Discretionary protection policies govern the access of users to the contents on the basis of the user's identity and the authorizations she possesses. Discretionary methods must deal with the real flow of information in a system [18]. On the other hand, the mandatory aspect of MAC relies in that users access objects through a MAC system and are not able to change or reassign access rights. DAC allows an object's owner to determine the access rules for that object. While yielding more flexibility than MAC systems, DAC loses the ability to provide provable security to resources.

Generic access control models have been studied extensively in non-collaborative domains, providing the basic framework to describe protection systems such as classic access matrix models (subject-object-right) and reference monitors [2], among others. 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). Interested readers can find further details in [21].

Models based on access matrices 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. Furthermore, these approaches lack the ability to support dynamic changes of access clearances. For teamwork applications, some collaborative frameworks use roles in conjunction with ACLs, such as SUITE and Intermezzo frameworks [7].

Role-based access control (RBAC) methods were proposed as an alternative to the models based on users' identities. In RBAC policies, permissions are assigned to roles (job functions or responsibilities) rather than to individual users. In the case of collaborative environments, it is insufficient to have role permissions based on object types.

Some cryptographic-based mechanisms have been suggested to solve the problem of content distribution, such as Broadcast Encryption [10]. This is a cryptographic technique for implementing compliant authorized domains, and can be used as a replacement for public key cryptography in certain applications.

A distributed access control model is addressed in [14] 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. A 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 do and can not. A possible solution would be that the delegate should act in his own name, not in hers.

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. Furthermore, sharing can be encouraged by imposing a cost on the downloads. 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 these attributes by the issuer.

We have established some requirements that an access control model for collaborative environments should support: user clearances and content security labels. Our proposed model 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.

3. Assumptions and notation

The main purpose of this section is to establish some basic notation and discuss a few working assumptions. Throughout this work, we will assume the following working (operational) hypotheses:

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 provider. After several transactions with different providers, it is expected that a node will have a "portfolio" of authorization certificates. Apart from these, each participant has a unique identifier (pseudonym, IP address, network name, etc). By now, anonymity is not desired. Identification of contents is also required. A unique name, which is also used for searching the content, is associated with the content.
2. *Digital signatures* cannot be forged unless the attacker gets access to private keys. Anyone can verify the authenticity of a node's signature by applying the Byzantine fault-tolerant public-key authentication protocol presented in [17].
3. *Reliable networking*. Protocols at network and transport layer provide secure and reliable communication among nodes. This is especially important in the case of M-P2P applications, which typically will execute on wireless, ad hoc networks.

   Next we summarize the notation used in this paper:

   - N is the number of network nodes. Each node is denoted by n_i. Specific nodes will be occasionally designated by capital letters: A, B,...
   - Each node n_i has a pair of public and private keys, denoted by K_i and K_i^{-1}, respectively. In turn, $enc_K(x)$ represents the encryption of message x using K as key.
   - m denotes a content that a node wish to publish, and $h(m)$ represents the result of applying a cryptographic hash function on m.
   - $s_i(x)$ is n_i's signature over x, i.e.:

     $$s_i(x) = enc_{K_i^{-1}}(h(x))$$

     whereas $s_i^j(x)$ is n_i's signature over x concatenated with j's identity, i.e.:

     $$s_i^j(x) = enc_{K_i^{-1}}(j\|h(x))$$

   - $\mathcal{P}(d,x)$ represent a d-difficult cryptographic puzzle ($d \in [0,\ldots,d_{\max}]$) whose solution is x.
   - C_m is a content certificate associated with m.
   - $C_{n_i}^{n_j}$ is an authorization certificate issued by n_j to n_i.

The specific structure of puzzles, content certificates and authorization certificates will be clarified later.

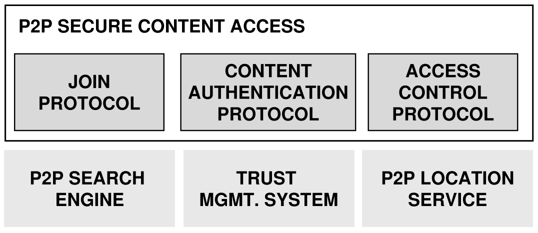


Fig. 1. Main building blocks.

4. An authentication and access control protocol

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, and Secure Content Access subprotocol (see Fig. 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 *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 our scheme, the system always transmits and presents contents encrypted by the sources by using some symmetric cipher and a key K_S. Furthermore, the content is accompanied by a certificate, C_m, which will serve both to guarantee m's authenticity and to dictate the security clearance required to access m. The details of this certificate and a full description of the scheme are provided below. The basic idea is that the content certificate contains a puzzle whose solution is K_S, i.e. the key required to decipher $enc_{K_S}(m)$. Basically, we may use a trapdoor function to supply collaborative requesters with l-bits out of the total bits of the key. These l bits can be seen as an advantage for honest peers.

Providers control access to their contents by means of authorization certificates, which includes the requester clearance L_i. The clearance represents the trustworthiness of a holder not to disclose non-authorized content. On the other hand, puzzles' difficulty can be set by providers (e.g. the maximum $d_{\max}$) according to the security level, L_i, associated with each content. In most situations, the credential emitted to new users will have the lowest security clearance.

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.

Briefly, we can identify three different possible scenarios after downloading the encrypted content. If user B has a security clearance issued by A and with level equal or higher than L_i, then B can use it to easily solve the puzzle. On the contrary, if B's clearance is not enough, the difficulty of solving the puzzle (and therefore decrypting the content) will not get reduced. In this case, the provider A can 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, or simply mount a brute force attack on the (probably very hard) puzzle, whose complexity will depend on the length of K_S.

In any of the three scenarios, once B gets access to m, she must verify its authenticity by means of the Content Authentication subprotocol. Next we describe in detail each of the scheme's subprotocols.

4.1. 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 labels, such as the classic set "confidential", "restricted", "secret", and "top-secret", but can also be extended to support additional usage classifications depending on the owner's desires. As m is always published encrypted using a (symmetric) cipher with key K_S, B needs an authorization certificate issued by A to get access to m. Peers who successfully obtain this authorization certificate benefits from a trapdoor information for a faster key recovery process, and therefore the content decryption takes less time and fewer resources.

Content owners must keep a local (and private) database containing the keys used to encipher contents and the secret trapdoor values for each content's security label, as shown in Table 1.

Next 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 [3]. Thus,

Table 1
Table of puzzles maintained by each content provider

Sec. label	Session key	Trapdoor
L_1	K_{S_1}	$secret_1$
$\vdots$	$\vdots$	$\vdots$
L_n	K_{S_n}	$secret_n$

B, who is still unknown by A, must send some information to A:

$$B \rightarrow A : m1 = \langle B, A, RF, s_B(B, A, RF)\rangle$$

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.

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 issued by a content owner for granting access to a specific requester, with a limited validity period and a security label L_i. How owners evaluate requesters' intentions will be discussed in Sections 5 and 6.

In case of acceptance, A generates and sends an authorization certificate "offer" C_B:$\langle B,A,t_e,L_i\rangle$ for B:

$$A \rightarrow B : C_B, s_A(C_B)$$

Upon reception, B will check A's signature, the expiration time t_e and the security clearance L_i provided. If this offer satisfies B, then she will return to A her signature on C_B:

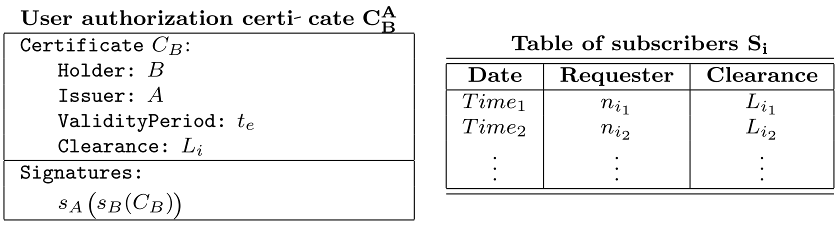
$$B \rightarrow A : C_B, s_B(C_B)$$

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

$$C_B^A : C_B, secret_{L_i}, s_A(s_B(C_B))$$
$$A \rightarrow B : enc_{K_B}(C_B^A), s_A\big(enc_{K_B}(C_B^A)\big)$$

Summarizing, A generates a certificate C_B^A for node B, containing (see Fig. 2):

- The identity of the authentication certificate owner (B), also called "holder", who establishes who has generated the content and is its legitime owner.
- The identity of the issuer (A).
- Expiration date (t_e) of C_B.
- B's security clearance, L_i.
- Finally, the previous items are signed by the issuer A and the requester B (as shown in Fig. 3).
- In the same way, the trapdoor secret corresponding to L_i, $secret_{L_i}$, is associated with this authorization certificate.



User authorization certi- cate C_B^A

```
Certificate C_B:
    Holder: B
    Issuer: A
    ValidityPeriod: t_e
    Clearance: L_i
Signatures:
    s_A(s_B(C_B))
```

Table of subscribers S_i

Date	Requester	Clearance
$Time_1$	n_{i_1}	L_{i_1}
$Time_2$	n_{i_2}	L_{i_2}
$\vdots$	$\vdots$	$\vdots$

Fig. 2. User authorization certificate and table of subscribers.

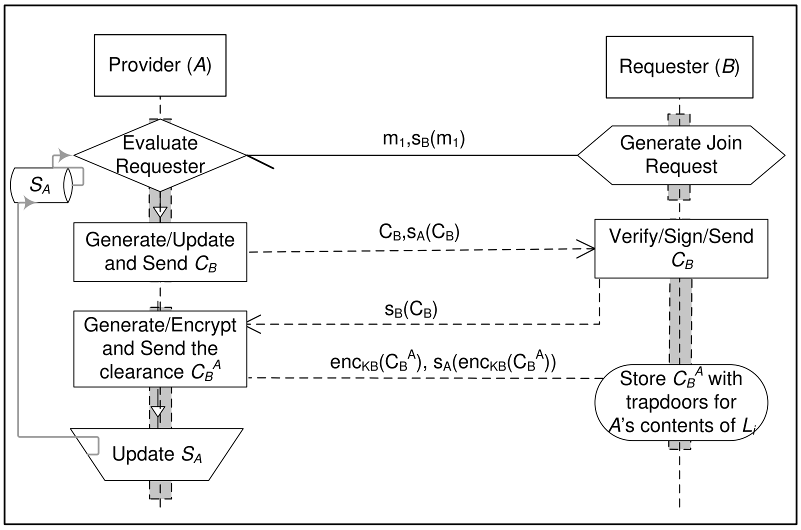

Provider (A)
Requester (B)
Evaluate Requester
m1,sB(m1)
Generate Join Request
SA
Generate/Update and Send CB
CB,sA(CB)
Verify/Sign/Send CB
sB(CB)
Generate/Encrypt and Send the clearance CB^A
encKB(CB^A), sA(encKB(CB^A))
Store CB^A with trapdoors for A's contents of Li
Update SA

Fig. 3. Join subprotocol.

Moreover, each provider must store successfully members or "subscribers" and their updated clearances, as shown in Fig. 2.

Since C_B^A includes the security label 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. 3.

4.1.1. 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 updating 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) [4]. The TMS incurs a fixed cost of having to store and exchange all behavior feedback items to generate a reputation value for each requester. Essentially, higher clearances are granted according to the past behavior of each peer. This way, our scheme can be successfully integrated with a TMS, for providers can use it in order to classify peers.
- Using proofs of work. In this case, the cost of getting a clearance for any node will depend on the time required for solving a puzzle of a selected level from a provider's list of challenges.
- Using a discretional policy. In this case, each node discretionary assigns clearance according to her personal criteria.

In any case, when a node has more than one clearance, her certificate must also include previous trapdoor values in order to access lower-security contents. For this, the field corresponding to the secret should be implemented as a list instead of as a single value:

	L_1	$\cdots$	L_x
secrets:	$secret_1$	$\cdots$	$secret_x$

4.1.2. Cryptographic puzzles

There exists many ways of constructing a function such that its computation is quite difficult unless a party possesses some piece of (secret) additional information. 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 is published. Upon knowing a trapdoor value (e.g. a number of bits of the key), the effort required to recover x can be adjusted from very hard to a quite difficult problem according to the number of bits revealed in the trapdoor value.

For instance, if a 256-bits key is used to encipher message 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 value of x. On the other hand, if we want the party to devote more resources on the computation for accessing x, a lower informative trapdoor value should be used. For example, by providing 128 correct bits of the key, we force her to carry out around 2^{127} operations. This way, the number of revealed bits can be seen as the difficulty parameter in the cryptographic puzzle $\mathcal{P}(d, x)$.

Apart from this, there exist other possibilities to use cryptographic primitives for building up similar puzzles (e.g. hash functions in which a preimage of a given value must be found).

4.2. Content authentication subprotocol

The main objective of this subprotocol is to maintain content integrity, ensuring its authenticity and avoiding non-authorized content alterations. This is achieved through the collaboration of a fraction of peers in the system. Let A be the legitime owner of a given content m. Previously to content distribution, A will generate a content certificate C_m. For this, A first selects a subgroup of signing nodes $\mathcal{S}$, with $|\mathcal{S}| = k$, among her group of trusted members. The way in which this selection is carried out is not discussed here, but it could depend, for example, on their availability and trust level. C_m (Fig. 4 (left)) is structured in three parts as follows:

- The certificate C, containing the following fields: The identity of the originator, which ultimately establishes who has generated the content and is its legitime owner, the identity and hash of the content, $h(m)$, assuring its integrity, and the ordered list of signers (OLS). The latter contains the identity of $k+1$ network nodes, denoted by $n_0, n_1, \ldots, n_k$, where $n_0 = A$ is the content originator.
- Three attributes: a valid period for C_m, the security label of the content, L_m, and the puzzle $\mathcal{P}(d_{\max}, K_S)$. Puzzle's solution is the decryption key for obtaining m.
- Finally, the previous items are recursively signed by the nodes listed in the OLS. First, A signs the certificate. The resulting signature is subsequently checked and signed by n_1, and so on.

C_m must be progressively signed by the nodes included in the OLS. We have identified two different ways in which this process can be carried out. In a distributed approach, each node is responsible of sending the signed certificate to the next one in the list. In this way, A simply sends the initial certificate, C_0, to the first signer, n_1, and waits until C_m arrives (as shown in Fig. 5 (left)). 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

Content certificate $\mathbf{C_m}$

Certificate C: Holder: A ID: I_m Content: $h(m)$ OLS: $A, n_1, \ldots, n_k$
Attributes: Validity Period: t_e Sec. Label: L_m Puzzle: $\mathcal{P}_m = \mathcal{P}(d_{max}, K_S)$
Signatures: $s_{n_k}(\cdots(s_{n_1}(s_A(C))))$

Table of signed certificates $\mathbf{T_i}$

Date	Certificate received	Signature $\mathbf{s_{n_i}}$
$Time_1$	C_{m_1}	s_{i1}
$Time_2$	C_{m_2}	s_{i2}
$\vdots$	$\vdots$	$\vdots$

Fig. 4. Content certificate and local database maintained by each certification node.

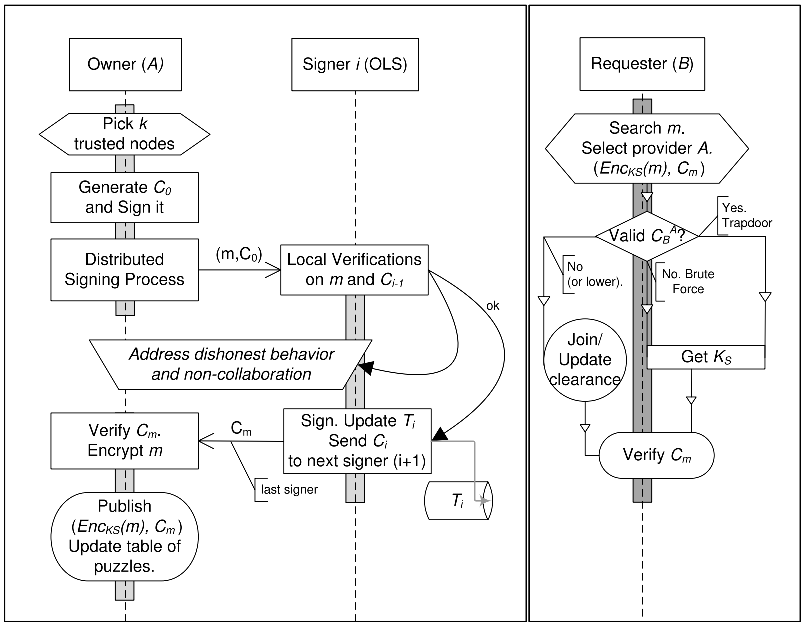

Owner (A)
Signer i (OLS)
Requester (B)
Pick k trusted nodes
Generate C0 and Sign it
Distributed Signing Process
(m,C0)
Local Verifications on m and Ci-1
ok
Address dishonest behavior and non-collaboration
Verify Cm. Encrypt m
Cm
Sign. Update Ti Send Ci to next signer (i+1)
last signer
Ti
Publish (EncKS(m), Cm) Update table of puzzles.
Search m. Select provider A. (EncKS(m), Cm)
Valid CB^A?
Yes. Trapdoor
No (or lower).
No. Brute Force
Join/ Update clearance
Get KS
Verify Cm

Fig. 5. Content authentication and access subprotocols.

process. As a centralized alternative, A could be 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.

Signers have to perform a *local verification* stage in order to ensure that they are not being cheated on. We will denote by $C_0, C_1, \ldots, C_k = C_m$ the successive versions of the certificate as it passes through the list of nodes. First, each $n_i \in \mathcal{OLS}$, when it receives certificate C_{i-1}, should verify the authenticity and integrity of m (computing $h(m)$ and comparing the result with the value contained in C_m), verify the signatures contained in the received certificate according to the list order (instancing Pathak and Iftode's public key authentication protocol if necessary), as well as to authenticate the originator (obtaining A's public key if unknown). 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, $s_{n_i}(C_{i-1})$ on a local table of signatures, denoted T_{n_i} (Fig. 4 (right)).

Finally, A publishes the content m encrypted with the session key K_S (which should never be distributed in clear), along with the content certificate:

$$(enc_{K_S}(m), C_m)$$

Fig. 5 (left) depicts graphically the content authentication subprotocol.

4.3. Content access subprotocol

Let B be the requester who wish 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. The results should contain at least the content descriptor and 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 $enc_{K_S}(m)$, as explained previously.

Now 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. Recall that successful past transactions with a well-known provider mean that B already has an authorization certificate (access right) to use A's offered services. The possession of a valid C_B^A will give several advantages to B, particulary if the clearance of B is high enough, i.e. $L_B \geqslant L_m$. In this case, B can use the information enclosed into her authorization certificate to easily solve the puzzle included in the content certificate C_m. This process provides B with K_S, the key required to decrypt m.

In case of B 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.
2. Try to update or increase her clearance (see Section 4.1.1).
3. Try to solve the maximum hardness puzzle $\mathcal{P}(d_{\max}, K_S)$ included in C_m for getting m.

Whatever the case may be, B finally gets K_S and decrypts the desired content. Now she must verify the correctness of C_m in order to ensure the authenticity and integrity of m. For this, B performs the following steps:

- *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 verifies the concatenated signatures of 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 TMS [19].

In Fig. 5 (right) it is graphically shown the content access subprotocol. Finally, a summary of the entire scheme is shown in Fig. 6.

5. Security analysis

In this section, we provide an informal analysis concerning the correctness of our proposal in terms of security. For this, we discuss several attack scenarios and forms of malicious behavior which can occur during each phase of the protocol execution, showing how the scheme is prevented against them.

5.1. Join subprotocol

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

1. *Eavesdropping*. An attacker cannot listen the four messages transmitted during the Join subprotocol, for all the communication between A and B are protected by the network/transport layer (see assumptions in Section 3). This can be achieved by using e.g. Secure Socket Layer (SSL), or any other scheme allowing parties to mutually authenticate and establish a session key under which all the communication is encrypted.
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 secure channel established between both parties prevent this situation to occur. Such a modification can only lead to an incorrect message authentication in the lower layers, but never to any gains for the attacker.

```
Join

    $B \rightarrow A : B, A, RF, sig_B(B, A, RF)$
    $A \rightarrow B : C_B, sig_A(C_B)$
    $B \rightarrow A : sig_B(C_B)$
    $A \rightarrow B : enc_{K_B}(C_B^A), sig_A\left(enc_{K_B}(C_B^A)\right)$

Content authentication

    1. $C_m$ generation:
       $A$ generates $C =< A, I_m, h(m), OLS >$ and signs:$C_0 =< C, s_A(C) >$
    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 $C_i$
          (c) $n_i$ updates $T_i$ with the tuple $< timestamp, C_{i-1}, s_{n_i}(C_{i-1}) >$
          (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 $\left(enc_{K_S}(m), C_m\right)$

Content access

    Case 1:$B$ has an appropriate $C_B^A$:
          1.$B$ uses $C_B^A$ for easily solving $\mathcal{P}(d_{max}, K_S)$.
          2.$m$ is decrypted by using $K_S$ (puzzle solution).
    Case 2:$B$ has no valid authorization:
          1.Initiate Join subprotocol, or
          2.Try to update/increase her clearance, or
          3.Try to solve the maximum hardness puzzle $\mathcal{P}(d_{max}, K_S)$
    $C_m$ verification:
          1.$h(m)$ comparison
          2.$OLS$'s public keys verification (if not done before)
          3.Verification of $C_m$'s chain of signatures
```

Fig. 6. Summary of the proposed content authentication and access control scheme.

3. *Message reply attacks*. An attacker first listens the messages exchanged between *A* and *B* and then tries to reproduce the session by using some of them. The protocol is robust against such a situation for two reasons. First, the underlying secure communication protocol will detect that the messages are encrypted under a different session key. Second, the identities of of both *A* and *B* are included in the messages, 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 between *A* and *B*. However, she is unable to do that for the same reason previously described for the case of *message modification attack*.
5. *Message dropping attacks*. Given enough control over the network infrastructure, an attacker can try to delete some of the messages exchanged between *A* and *B*. 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.
6. *Impersonation attacks*. Assume that an attacker can success in hijacking a secure session between *A* and *B*. 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 either *A* or *B* trying to exhibit malicious behavior against the other party.
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 $K\prime_{n_i}$ when it is not. This kind of man-in-the-middle attack is detected if at least one peer (apart from *A*) is honest.

5.2. Content authentication subprotocol

The discussion provided above for attacks 1–7 is applicable to the content authentication protocol in all the aspects concerning an attacker trying to influence over the secure communication. Apart from those, some other questions arise in this case:

1. *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 origi-

nal 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.

2. *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: $B, n_{1'}, \ldots, n_{k'}$ SignAlgorithm: $Algorithm$
Attributes: Validity Period: t'_e Sec. Label: $L_{i'}$ Puzzle: $\mathcal{P}'_m = \mathcal{P}(d'_{max}, K'_S)$
Signatures: $s_{n_{k'}}(\cdots(s_{n_{1'}}(s_B(C'))))$

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.

3. *No cooperation*. Cooperation is crucial during the content certificate generation stage. 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, 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 it is done in some current P2P systems for encourage file sharing. Nevertheless, this is something external to our proposal. Furthermore, 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.

5.3. Content access subprotocol

The search and downloading process is external to our proposal and provided by a P2P file-sharing application. Since the contents are distributed encrypted, the protocol does not intervene in this process. Once the content has been successfully downloaded, there are several scenarios for an attacker:

1. *Breaking cryptographic puzzles*. At the content access stage, peers without the proper authorization (e.g., clearance) can choose to break the puzzle. The only way of breaking it is by using a brute force attack and trying each possible key until the content is correctly decrypted. Assuming a correct implementation of the encryption process and an appropriate length for the keying material, completing successfully this task falls out of the resources of an attacker.
2. *Manipulation of authorization certificates*. An user B with an insufficient clearance cannot upgrade it exclusively by herself, for the key bits required to do so are only known by the content's owner A. Therefore, the upgrading process can only be done by A through the Join 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.

5.4. 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.

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.

6.1. Computational effort

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

Table 2 presents the content access life cycle summarizing the time sequence, the number of cryptographic operations and the complexity for each stage of the protocol. Furthermore, we include in Table 3 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 and content access subprotocol.

Fig. 7 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. Content certificate subprotocol 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 a symmetric encryption and a puzzle generation.

For instance, generating a certificate for contents between 1 and 10 MB takes less than 9 min 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 20 signers takes approximately 1 h. See, however, that this task is carried out just once, and that content access is considerably faster.

6.1.1. 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 an authorization certificate is around 600 bytes, depending on the security level issued to the user and the signing algorithm chosen. In the case of content certificates, the size is similar.

Content providers and signing nodes must maintain a number of local databases (tables $\mathbf{Z}_i$, $\mathbf{T}_i$ and $\mathbf{S}_i$). 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.

Table 3
Speed of cryptographic primitives used in our simulations [1]

Algorithm	ms/operation
`RSA 2048 Encryption`	0.45
`RSA 2048 Decryption`	28.41
`RSA 2048 Signature`	28.13
`RSA 2048 Verification`	0.45
	MB/second
`AES-256`	48.229
`SHA-256`	44.460

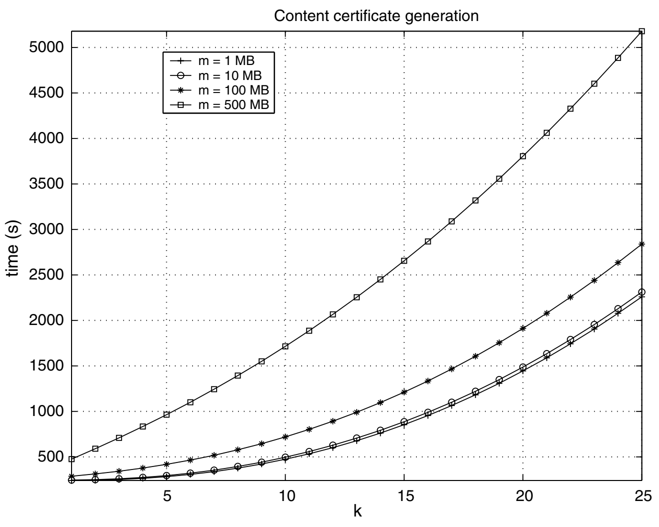
6.2. Communication overhead

Next we analyze efficiency according to the communication overhead imposed by the proposal. Table 4 presents the number of messages and the complexity in terms of communication transmission for each subprotocol stage. For example, Join process linearly increases slowly as shown in Fig. 8. This figure 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's length and the complexity of the challenges in Pathak and Iftode's messages. On the other hand, the upper bound in our simulations is given by a content certificate generation with 25 signers: more than 350 messages.

Table 2
Efficiency analysis (computational effort)

Subprotocol	Stage	No. 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	$3S + 3V + 1E + 1D$	
	Subtotal:	$4S + 4V + 1E + 1D + 1PI$	
Authentication	2.1 C_m generation	$1H + 1S$	$\mathcal{O}(\mid m \mid k^3 \log k)$
	2.2 Signature process	$kS + kH + \frac{k(k+1)}{2}V + kPI$	
	2.3 Content publishing	$1E + 1Z_G$	
	Subtotal:	$(k+1)(H + S + \frac{k}{2}V) +$ $1E + 1Z_G + kPI$	
Access	3.1 Access Request	$1Z_S + 1D$	$\mathcal{O}(\mid m \mid k^2 \log k)$
	3.2 B's Checking on C_m	$1H + kPI + kV$	
	Subtotal:	$1Z_S + 1D + 1H + k(PI + V)$	

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; Z_S, puzzle solving; Z_G, puzzle generation.



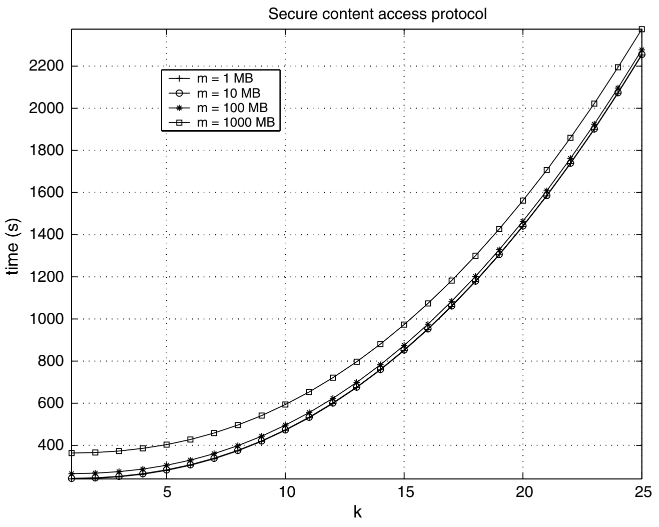


Fig. 7. Computational cost (worst case) for content certificate generation and content access subprotocol.

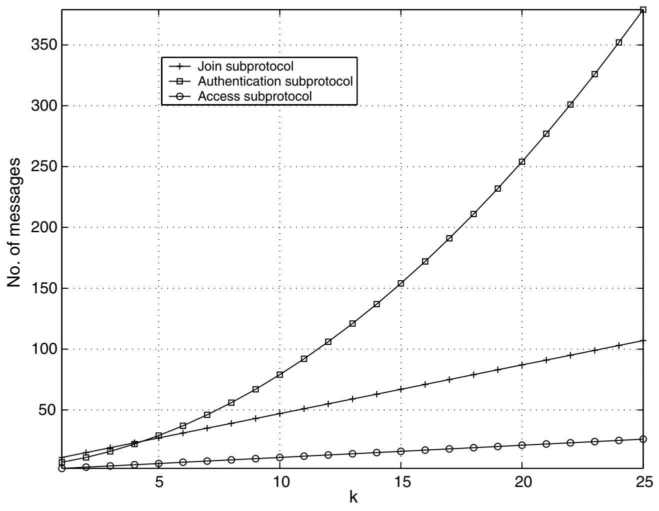


Fig. 8. Communication cost (number of messages transmitted in the worst case) for each subprotocol.

Table 4
Efficiency analysis (communication overhead)

Subprotocol	Stage	No. messages	Complexity
Join	1.1 B's Request	1	$\mathcal{O}(k)$
	1.2 A's checking	$4k+1$	
	1.3 C_B^A generation	$3+2$	
	Subtotal:	$4k+7$	
Authentication	2.1 C_m generation	$1+1$	$\mathcal{O}(k \log k)$
	2.2 Signature process	$2k+\frac{k(k+1)}{2}$	
	2.3 Content publishing	$1+1$	
	Subtotal:	$2k+4+\frac{k(k+1)}{2}$	
Access	3.1 Access Request	1	$\mathcal{O}(k)$
	3.2 B's Checking C_m	k	
	Subtotal:	$k+1$	

7. Conclusions and future work

Even in a mobile environment, a P2P content distribution scheme can be more efficient and economical than a traditional client–server scheme. However, security issues – in particular authorization and authentication – must be adapted to the specific nature of decentralized environments and mobile nodes.

In this paper, we have introduced an access control 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. Content owners can control who accesses their contents by discretionally issuing clearances in the form of authorization certificates.

Our motivating scenarios are the implementation of these concepts in collaborative working environments, supported by mobile ad hoc networks and file-sharing applications. Our scheme represent a secure mechanism for providing guarantees that a content is authentic and has not been altered, even if it is a replica of the original and its source has lost control over it.

Our future work include several research lines. First, it would be interesting to measure how serious the problem of fake content distribution is in real P2P environments, such as BitTorrent, eMule and eDonkey systems, to name a few. A second line is related to the study of specific ways according to which the clearance granting/updating process can be integrated with a 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.

In principle, the process for generating content certificates may be performed by using a multisignature scheme. This is generally a more efficient way to gather up a number of signatures over a given document than by doing it sequentially. Finally, we will investigate ways to extend the authentication and access control services here presented to more complex capabilities, such as privilege delegation and revocation.

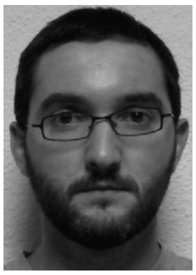
Acknowledgments

We would like to express our gratitude to the anonymous reviewers for the insights and comments provided during the review process, which have greatly contributed to improve the quality of the original manuscript.

References

[1] Available from: <http://www.eskimo.com/~weidai/benchmarks.html>.
[2] M. Abadi, M. Burrows, B. Lampson, G. Plotkin, A calculus for access control in distributed systems, ACM Transactions on Programming Languages and Systems (TOPLAS) 15 (4) (1993) 706–734.
[3] K. Aberer, M. Hauswirth, An overview on peer-to-peer information systems, in: Proceedings of Workshop on Distributed Data and Structures, Paris, France, March 2002, pp. 171–188.
[4] W. Adams, N. Davis, Tms: a trust management system for access control in dynamic collaborative environments, in: Proceedings of the 25th International Conference on Performance, Computing, and Communications, IEEE, Arizona, USA, April 2006.
[5] S. Capkun, L. Buttyán, J.-P. Hubaux, Self-organized public key management for mobile ad hoc networks, IEEE Transactions on Mobile Computing 2 (1) (2003) 52–64.
[6] J. Douceur, The sybil attack, in: Proceedings of the 1st International Workshop on Peer-to-Peer Systems, Cambridge, USA, March 2002, pp. 251–260.
[7] W. Edwards, Policies and roles in collaborative applications, in: Proceedings of the ACM Conference on Computer Supported Cooperative Work, ACM, ACM Press, Boston, Massachusetts, November 1996, pp. 11–20.
[8] S. Farrell, R. Housley. An internet attribute certificate profile for authorization. *RFC 3281*, April 2002.
[9] P. Fenkam, S. Dustdar, E. Kirda, G. Reif, 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, IEEE Computer Society, Pittsburgh, USA, June 2002, pp. 95–102.
[10] A. Fiat, M. Naor, Broadcast encryption, in: Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology, California, USA, August 1994, pp. 480–491.
[11] J. Hubaux, L. Buttyán, S. Capkun, The quest for security in mobile ad hoc networks, in Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking and computing, Long Beach, USA, 2001, pp. 146–155.
[12] A. Juels, J. Brainard, Client puzzles: a cryptographic defense against connection depletion attacks, in: Proceedings of the Networks and Distributed Security Systems, California, USA, February 1999, pp. 151–165.
[13] L. Lamport, R. Shostak, M. Pease, The byzantine general problem, ACM Transactions on Programming Languages and Systems 4 (3) (1982) 382–401.
[14] J. Lopez, R. Oppliger, G. Pernul, Authentication and authorization infrastructures (aais): a comparative survey, Computer Communications 27 (16) (2004) 1608–1616.
[15] P. Maniatis, T. Giuli, M. Roussopoulos, D. Rosenthal, M. Baker, Impeding attrition attacks in p2p systems, in: Proceedings of the 11th ACM SIGOPS European Workshop, ACM, Leuven, Belgium, September 2004.
[16] E. Palomar, J. Estevez-Tapiador, J. Hernandez-Castro, A. Ribagorda, Certificate-based access control in pure p2p networks, in: Proceedings of the 6th International Conference on Peer-to-Peer Computing, IEEE, Cambridge, UK, September 2006, pp. 177–184.
[17] V. Pathak, L. Iftode, Byzantine fault tolerant public key authentication in peer-to-peer systems. Computer Networks. Special Issue on Management in Peer-to-Peer Systems: Trust, Reputation and Security, 50(4):579–596, March 2006.
[18] R. Sandhu, 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, ACM, Stockholm, Sweden, June 2005, pp. 147–158.
[19] A. Selcuk, E. Uzun, M. 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, Chicago, USA, April 2004, pp. 251–258.
[20] Q. Tang, K. Choo, Secure password-based authenticated group key agreement for data-sharing peer-to-peer networks, in: Proceedings of the 4th International Conference on Applied Cryptography and Network Security, Singapore, June 2006.
[21] W. Tolone, G. Ahn, T. Pai, S. Hong, Access control in collaborative systems, ACM Computing Surveys 37 (1) (2005) 29–41.
[22] X. Zhang, S. Chen, R. Sandhu, Enhancing data authenticity and integrity in p2p systems, IEEE Internet Computing (2005) 42–49.

Esther Palomar is Assistant Professor at the Computer Science Department of Carlos III University of Madrid. She holds a M.Sc. in Computer Science (2004) from Carlos III University of Madrid. Before joining the university, she worked as forensic analyst for a well-known firm in the field. Currently, she belongs to the Cryptography and Information Security Group, where she is performing her Ph.D. studies. Her research interests are focused on security in peer-to-peer and ad hoc networks.



Juan M. Estevez-Tapiador is Associate Professor at the Computer Science Department of Carlos III University of Madrid. He holds a M.Sc. in Computer Science from the University of Granada (2000), where he obtained the Best Student Academic Award, and a Ph.D. in Computer Science (2004) from the same university. His research is focused on cryptography and information security, especially in formal methods applied to computer security, design and analysis of cryptographic protocols, steganography, and some theoretical aspects of network security.

Julio C. Hernandez-Castro is Associate Professor at the Computer Science Department of Carlos III University of Madrid. He has a B.Sc. in Mathematics, a M.Sc. in Coding Theory and Network Security, and a Ph.D. in Computer Science. His interests are mainly focused in cryptology, network security, steganography and evolutionary computation.

Arturo Ribagorda is Full Professor at Carlos III University of Madrid, where he is also the Head of the Cryptography and Information Security Group and currently acts as the Director of the Computer Science Department. He has a M.Sc. in Telecommunications Engineering and a Ph.D. in Computer Science. He is one of the pioneers of computer security in Spain, having more than 25 years of research and development experience in this field. He has authored 4 books and more than 100 articles in several areas of information security. Additionally, he is member of the program committee of several conferences related to cryptography and information security.