Certificate-based Access Control in Pure P2P Networks

Esther Palomar, Juan M. Estevez-Tapiador, Julio C. Hernandez-Castro, Arturo Ribagorda
Computer Science Department – Carlos III University
Avda. Universidad 30, 28911 Leganes, Madrid
{epalomar, jestevez, jcesar, arturo}@inf.uc3m.es

Abstract

Pure Peer-to-Peer (P2P) networks are characterized as being extremely decentralized and self-organized, properties which are essential in a number of environments, including teamwork, collaborative, and ad-hoc systems. One of the features offered by P2P networks is the possibility of having several replicas of the same content distributed among multiple nodes. Despite its advantages (e.g. robustness and fault tolerance), it is crucial to guarantee content authenticity, as well as to enforce appropriate access control policies. 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. In a previous work, we presented a protocol for content authentication based on public key certificates that does not rely on the existence of a public key infrastructure. In this paper, we show how these certificates can be extended to provide authorization capabilities. In our scheme, each peer classifies her contents according to several security labels. Peers allowed to access a given content must have a security clearance of at least the same level that the content's. These security clearances, which take the form of attributes in public key certificates, can be discretionally issued by the content provider.

1 Introduction

One of the features offered by 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), it is crucial to guarantee properties such as content authenticity, as well as to enforce appropriate access control policies. Furthermore, many inherent characteristics of collaborative environments introduce new requirements for access control: support for mobile devices, different applications, control decentralization, and off-line working. Authorization in such environments cannot be generally provided by means of existing models, which were developed for centralized systems. For instance, 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 trusted third parties (TTP).

The most accepted solution for authentication and authorization services in classic distributed environments relies on the existence of a Public Key Infrastructure (PKI). Once that a public key can be securely associated to a given party, the integrity of any content generated by her can be ensured through her sign, 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.

Currently, content providers want to facilitate the emergence of new models that maintain some control over the file-sharing process in P2P networks. The objective of a content protection scheme is to raise the barrier for casual violations and to require a concerted effort by every requester. We explore further extensions that help reduce the effectiveness of dishonest behavior. In particular, sharing can be encouraged by imposing a cost on the downloads (e.g. resolving a cryptographic puzzle). Following this line, recent works have also studied how to use a P2P network to prevent Denial of Service (DoS) attacks on the Internet [8, 9]. Furthermore, a number of approaches have been designed that support secure authentication and authorization paradigms in distributed systems, such as P2P collaborative environments [6], data-sharing P2P networks [14], and mobile ad-hoc networks [3].

Moreover, content quality is an important issue for file sharing users. Currently, digital content protection tries to eliminate suspicions over non-authorized content modifications, thus avoiding the generation of mistrust among the community. Several researchers agree that in order to do so, requiring the attention of a human in order to perform a task before granting access to a service could be a good

idea. This task could also be implemented as a puzzle, even though this approach affects to the quality of service (connection availability, bandwidth, and storage) provided to a legitimate user. The use of computational puzzles to encourage fair resource allocation has been considered in [4]. Nevertheless, a critical keypoint here is that clients need to trust the system in providing them with the requested contents once they had solved the puzzle.

However, it is highly unrealistic to assume that every integrating node will always exhibit a honest behavior, even if they have systematically behaved correctly in the past. 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, we have presented a protocol for content authentication based on public key certificates that does not rely on the existence of a PKI [10]. Our content authentication protocol is based on the scheme presented in [11], which addresses the issues related to public key authentication in P2P systems based on Byzantine agreement. The proposed mechanism is autonomous and does not require a trusted third party (TTP).

In this paper, we show how digital certificates used for content authentication can be extended to provide authorization capabilities, much in the way a X.509 public-key certificate can be used as an attribute certificate. We propose a rational content access procedure based on proofs of computational effort. In our scheme, each peer classifies her contents according to several security labels. Peers allowed to access a given content must have a security clearance of at least the same level that the content's. These security clearances, which take the form of attributes in public key certificates, can be discretionally issued by the content provider.

The rest of this article is organized as follows. Section 2 briefly overviews some related work and the motivation of our solution. Section 3 introduces notation and a few working assumptions. In Section 4, we first present the basic operation of our scheme and subsequently discuss the three main subprotocols. Finally, Section 5 concludes the paper and outlines some future research directions.

2 Related work and motivation

There has been much research in computer applications for facilitating collaboration among multiple, distributed users. However, there has been relatively little work done in controlling access to the collaboration environment and shared data. Most collaborative systems give all participants the same rights to all objects and expect access to be controlled by a social protocol. Thus, they do not provide computer support for preventing mistakes, conflicting changes, or unauthorized access.

Generic access control models have been studied extensively in non-collaborative domains, which provide the basic framework to describe protection systems, such as classic access matrix models (subject-object-right) and reference monitors among others [1]. 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 [15]. Access matrix-based models have some weaknesses when applied in a collaborative environment due to the impossibility of relating access rights to content, 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 [5]. In Role-based access control (RBAC) methods, permissions are assigned to roles (job functions or responsibilities) rather than to individual users, but for collaborative environments, it is insufficient to have role permissions based on object types.

Some cryptographic-based mechanisms have been applied to solve the problem of content distribution, such as Broadcast encryption [7]. This is a cryptographic technique for implementing compliant authorized domains, and can be used as a replacement for public key cryptography in certain applications. It provides mechanisms for enforcing compliance without requiring device identification or authentication, while providing a reasonable level of control to limit massive content redistribution [12].

On the other hand, a similar approach to [11] is proposed in [13]. It presents an admission control system that mitigates Sybil attacks by adaptively constructing a hierarchy of cooperative admission control nodes. Implemented by the peer-to-peer nodes, the admission control system vets joining nodes via client puzzles. A node wishing to join the network is serially challenged by the nodes from a leaf to the root of the hierarchy. Nodes completing the puzzles of all nodes in the chain are provided a cryptographic proof of the vetted identity.

Balancing main features of collaboration and security, our solution establishes an access control scheme for pure P2P networks based on authorization certificates, 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. We have established some requirements that a generic 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

First, we introduce some notation and discuss a few working assumptions.

Throughout this work, we will assume the following two working (operational) hypotheses:

1. **Identification**. All participants need a certificate for accessing a desired content of a certain provider's directory. Each certificate is discretionally issued by the content provider. After several transactions with different providers, a node will posses a "portfolio" of authorization certificates. 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 [11].

Next we summarize the notation used in this paper:

- N is the number of network nodes. Each node is denoted by n_i. Eventually, specific nodes will be 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 the content that nodes wish to publish, whereas $h(m)$ represents 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 whose solution is x, and where $d \in [0, \ldots, d_{max}]$ is a parameter measuring the puzzle hardness.
- 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.

4 Access Control Scheme

In this Section, we first overview the full protocol operation by means of several scenarios. Our content authentication protocol based on Byzantine agreement is summarized in section 4.3. Finally, we present a cryptographic-puzzle-based scheme for a secure content download.

4.1 Basic operations

The proposed scheme, as illustrated in Figure 1, works as follows. We assume that the user B has at her disposal some kind of search mechanism to locate contents (Step 1). The usual situation is that, due to replication, the search engine returns a list of sources from which the content can be obtained (Step 2). Among the possible options, we will denote by A as the source selected by user B.

The content m is enciphered by the source by using some symmetric cipher and a key x. Furthermore, the content is accompanied by a certificate, C_m, which will serve both to guarantee m's authenticity and to dictate the required security clearance to access m. The details of this certificate and a full description of the scheme are provided below. The basic idea is that the certificate contains a puzzle whose solution is x, i.e. the key required to decipher $enc_x(m)$. The difficulty of this puzzle can be set by the content provided, though we assume it is the maximum allowed. i.e. d_{max}. Moreover, the content has an associated security level, L_i, also established by its provider.

At this point, there are several possible scenarios. If user B has a security clearance, issued by A, with a level higher than L_i, then B can show it to obtain a lighter puzzle (Case 4.a). In a different situation, user B could have a security clearance which is not enough to reduce the difficulty of solving the puzzle. In this case, the provider A can issue an easier puzzle or simply refuse the proposal (Case 4.b). Finally, if B has no security clearance issued by A, she must initiate the join subprotocol to obtain one. In most situations, the credential emitted to new users will have the lowest security clearance.

In any of the three scenarios, user B has to eventually solve the received puzzle to obtain the key x. Once she get access to the content m, she can verify its authenticity by means of the content authentication protocol described later. If this protocol successes, B is ensured that m has not been modified in an unauthorized way.

Next we describe in detail each subprotocol composing the scheme.

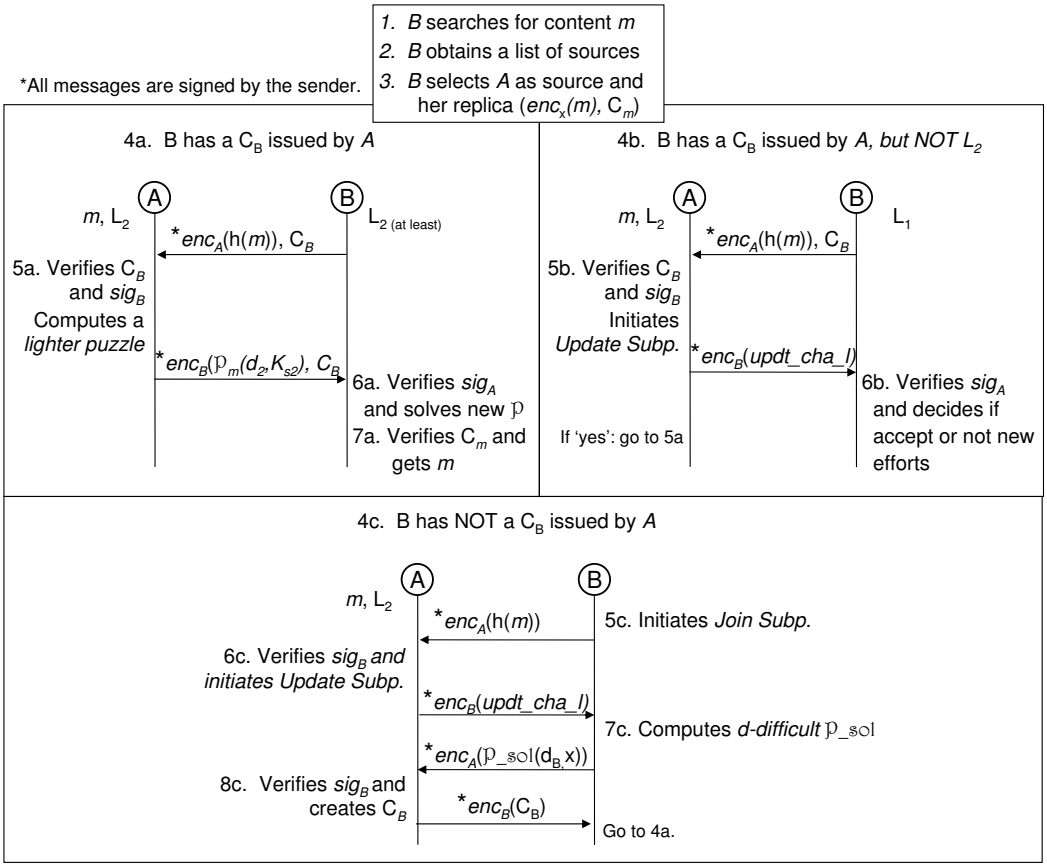

Figure 1. Overview of the scheme

4.2 Join Subprotocol

This stage is initiated when a new node joins the community or when the notion of security needs to be updated, i.e. when nodes can no longer be trusted. In the first case, when a node joins a community, it must be authorized by at least a node that is already part of the community. The process begins as follows: a content provider A offers a content m desired by a user B, who is still unknown by A. Thus, B needs an authorization certificate for being authorized to access m.

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 provider. At the first step, each user creates her public key and the corresponding private key, as some P2P file sharing systems [2]. Nevertheless, the authorization certificates are not issued neither by the owner (the requester), nor a CA. Authorization certificates are issued by a provider for granting access to a specific requester, with a limited validity period and a security label L_i. In the same way, each peer classifies her contents according to several security labels, such as "copy never", "copy once", "copy freely", and "copy no more", but can also be extended to support additional usage classifications.

Here, for simplicity, we assume that all certificates are issued with the same validity period. These certificates are small enough so that they can be stored on nodes and presented to providers in exchange of services. Thus, C_B^A includes the security authorization label which allows B accessing A's contents, should the former have the security clearance of at least the same level that the content's.

We explain some form of granting a clearance (through an authorization certificate) by a provider. The cost of joining the network for any legitimate node will depend on the time required for solving a puzzle of a selected level from a list of challenges.

To make this process fair, we need to fix the time it takes the average user to join the network. To do this, we first set the joining difficulty (measured in average time) to l. If a node must only solve a single puzzle of average time l, it is possible that it will be "lucky" and solve the puzzle on its

User authorization certificate C_B^A

Certificate C_B:
Subject: B
Provider: A
Valid Period: t_e
Sec. Clearance: L_i
Signatures:
$s_A\big(s_B(C_B)\big)$

Figure 2. User Authorization Certificate

first guesses. In fact, because the distribution of the time to solve the puzzle is uniform, the variance for the time taken to solve it is high:

$$\text{If } X \sim U(a,b) \qquad Var(X) = \frac{(b-a)^2}{12}$$

and hence unfair.

In summary, A generates a certificate C_B^A for node B, containing:

- The identity of the authentication certificate owner, who establishes who has generated the content and is its legitime owner.
- The identity of the provider.
- Expiration date (t_e) of C_B.
- B's security clearance L_i. This attribute represents the identity credential of a user according to her behavior. A possible classification could be Trusted, Confident, Novel, and Untrusted.
- Finally, the previous items are signed by the provider A and the requester B.

The structure of the certificate is summarized in Fig. 2.

4.3 Overview of our Content Authentication Subprotocol

The main objective of this scheme is avoiding non-authorized content alteration. Previously to content dissemination, the owner A of content m will generate a non-repudiable and unforgeable evidence ensuring the authenticity of m. For this, A first selects a subgroup (an appropriate value for the number k of signing nodes and their identities) $\mathcal{S}$, with $|\mathcal{S}| = k$, among her group of trusted parties. 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.

A generates a certificate C_m for content m, containing:

- The identity of the originator, which ultimately establishes who has generated the content and is its legitime owner.
- The identity of the content.
- A hash, $h(m)$, of m, assuring its integrity.
- 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. The nodes are selected among A's trusted group.
- Attributes: a valid period for C_m and the security class SC of m, principally. A possible classification of contents/services could be: TopSecret, Restricted, Confident, and Public.
- Puzzle: $\mathcal{P}(d_{max}, K_S)$. Its solution is the decryption key for obtaining m (very costly).
- 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.

The structure of the content certificate is summarized in Fig. 3(a).

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, each node n_i should perform a *local verification* stage when it receives certificate C_{i-1}. 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 $n_i \in \mathcal{S}$ should verify the authenticity and integrity of m, as well as to authenticate the originator. This includes obtaining A's public key, computing $h(m)$ and comparing it with the value contained in the received certificate. A crucial issue is that a node n_i must not sign twice the same content coming from different originators. For this, node n_i maintains a local table of signatures, denoted T_i, in which each signed message is stored in a new entry. Each entry registers the certificates previously signed: a timestamp, the received certificate (including the signatures contained), and the signature generated by the node. Table T_i's structure is shown in Fig. 3(b). Furthermore, 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.

If previous verifications succeed, the node adds its signatures to C_{i-1}, thus creating C_i. Each peer stores separately C_{i-1} and the generated signature, $s_{n_i}(C_{i-1})$, in her local database T_i.

We have identified two different ways in which the certificate can be signed by the nodes included in the OLS (see [10] for details). The main differences are the following:

Content certificate $\mathbf{C_m}$

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

(a)

Table of signed certificates $\mathbf{T_i}$

Date	Certificate received	Signature $\mathbf{s_{n_i}}$
$Time_1$	C_{m_1}	s_{i1}
$\vdots$	$\vdots$	$\vdots$
$Time_n$	C_{m_n}	s_{in}

(b)

Figure 3. (a) Content certificate; (b) local database maintained by each certification node

- In a distributed approach, each node is responsible of sending the signed certificate to the next node in the list. This way, A simply sends C_0 to n_1 and waits until C_m arrives. 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.

- 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. However, note that each node still has to perform its local verification stage in order to ensure that it is not being cheated on.

Finally, A publishes the content m encrypted with session key K_S, and the content certificate:

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

A will never distribute K_S, which is the puzzle solution. Cooperating users can opt to solve easier puzzles, depending on the security level included in the requester's authorization certificate.

4.4 Content Access Subprotocol

At the search stage, a requester node B first performs a search query, which typically leads to 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 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 C_m with all the information associated with the content, and $enc_{K_S}(m)$, as explained in the previous section.

First of all, and before verifying any signatures, B should select a source A within those returned by the query. This selection should be done taking into account the knowledge of some of the signers, or of the source, or, in the worst possible case, even at random. Recall successful past transactions with a well-known provider mean that B has an authorization certificate (access right) to use A offered services.

In any case, B must initiate this subprotocol once A is chosen, and the next stage depends on the following:

- If the authorization certificate is not delivered by A to B, or its expiration date has passed, or the desired resource's security class is higher than the requester's security level, then B could choose between two options: initiating the Join Subprotocol before executing the tasks at the item bellow, or directly trying to solve the maximum hardness puzzle $\mathcal{P}(d_{max}, K_S)$ included in C_m for getting m.

  In the first alternative, B signs an attribute published in C_m, $sig_B\big(enc_{K_A}\big(h(m)\big)\big)$, and sends it to A. Then, A can check B's identity and execute Join Subprotocol for creating a specific authorization certificate to B.

  In the other approach, B prefers breaking the puzzle published in the content certificate, getting the key for decrypting m. Puzzles can apply different cryptographic techniques and only brute force attacks can take the session key for granted.

- If B has an updated authorization certificate delivered by A. A must verify the authorization certificate related to the service it provides. Then, A can check B's identity and computes a new session key K_S to encrypt m. K_S is also sent to B by means of a new, weaker puzzle, $\mathcal{P}_m(d_s, K_S)$.

In both cases, once B gets the desired content, he must verify the correctness either of some or all of the concatenated signatures and their identities. At worst, B knows

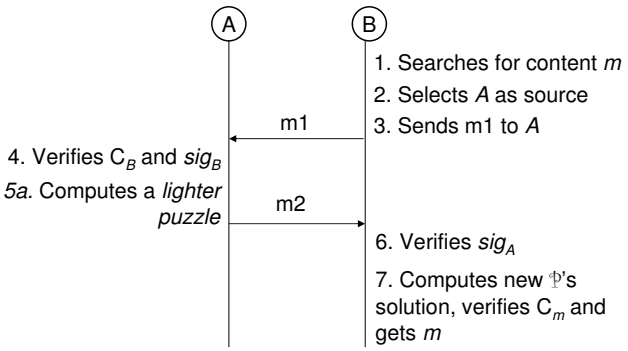

Figure 4. Content downloading sub-protocol

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). Since B can check the correct ending of the signatures verification process, then she can verify content's authenticity.

Once obtained m, the certificate should be checked to ensure the authenticity and integrity of m. For this, a node 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 obtains the public keys associated to each of the peers listed in the OLS. If any public key is unknown to B, it can be acquired by executing Pathak and Iftode's public key authentication protocol.

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

5 Conclusion and research directions

In this paper, we present an access control protocol especially oriented to pure P2P systems, which are environments mainly characterized by node transience and the lack of any centralized authority. Moreover, the proposed solution extends a P2P content authentication scheme based on Byzantine agreement, which allows for a secure content replication among peers maintaining integrity control over the published contents. Furthermore, our proposal includes a cryptographically-based countermeasure against non-authorized content alteration, in which proofs of computational effort may serve as a rational content access control.

On the other hand, content owners will control who access their contents, and have guarantees that requesters will make an effort for getting the desired content.

Our motivating scenarios are the implementation of these concepts in future P2P networks, collaborative environments, and file sharing applications in order to make them more reliable and secure. Furthermore, our scheme encourages users to share authenticated content, and represents 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.

References

[1] M. Abadi, M. Burrows, B. Lampson, and G. Plotkin. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 15(4):706–734, September 1993.

[2] 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.

[3] S. Capkun, L. Buttyan, 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.

[4] 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*, volume 740, pages 139–147, 1992.

[5] W. 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, ACM Press.

[6] 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. IEEE Computer Society.

[7] 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.

[8] 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.

[9] P. Maniatis, T. Giuli, M. Roussopoulos, D. Rosenthal, and M. Baker. Impeding attrition attacks in p2p systems. In *Proceedings of the 11th ACM SIGOPS European Workshop*, Leuven, Belgium, September 2004. ACM.

[10] E. Palomar, J. Estevez-Tapiador, J. Hernandez-Castro, and A. Ribagorda. A p2p content authentication scheme based on byzantine agreement. In *Proceedings of the International Conference on Emerging Trends in Information and Communications Security (ETRICS)*, pages 60–72. Lecture Notes in Computer Science. Springer-Verlag, June 2006.

[11] V. Pathak and 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.

[12] F. Pestoni, J. Lotspiech, and S. Nusser. xcp: Peer-to-peer content protection. *IEEE Signal Proccessing Magazine*, 21(2):71–81, March 2004.

[13] H. Rowaihy, W. Enck, P. McDaniel, and T. L. Porta. Limiting sybil attacks in structured peer-to-peer networks. Technical report, Network and Security Research Center, Department of Computer Science and Engineering, Pennsylvania State University, USA, July 2005.

[14] Q. Tang and 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.

[15] W. Tolone, G. Ahn, T. Pai, and S. Hong. Access control in collaborative systems. *ACM Computing Surveys*, 37(1):29–41, March 2005.