A Protocol for Secure Content Distribution in Pure P2P Networks

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

Abstract

A significant challenge for Peer-to-Peer (P2P) systems is maintaining the correctness and consistency of their global data structures and shared contents as peers independently and unpredictably join and leave the system. In such networks, it is necessary that some security mechanisms will be applied with the aim of avoiding attacks based on non-authorized content modifications. In this paper, we propose a content authentication protocol for pure P2P networks. The scheme also incorporates a rational content access procedure based on proofs of computational effort. Our proposal relies on a set of peers playing the role of a certification authority, since it is unrealistic to assume that appropriate trusted third parties can be deployed in such environments.

1 Introduction

In a Peer-to-Peer (P2P) file sharing system, peers communicate directly with each other to exchange information and share files. This kind of systems typically contain millions of dynamic peers involved in the process of sharing and collaboration without a central authority. Thus, P2P systems are characterized as being extremely decentralized and self-organizing. Specifically, these properties are essential in collaborative and ad-hoc environments, in which dynamic and transient population prevails.

Among the advantages of P2P networks, we emphasize the property of robust fault tolerance. Nevertheless, there are also some disadvantages, e.g. those derived from their decentralized nature, such as the complexity of network management and the existence of new security vulnerabilities. So far, security efforts have focused on anonymity, access control, integrity, and availability [3]. New areas are also being explored, such as fairness and authentication [2]. A recent paper [10], addresses the issues related to key authentication in P2P systems. This work describes a public key authentication method for P2P systems based on Byzantine agreement. The proposed mechanism is autonomous and does not require a trusted third party (TTP). Due to its relevance to the protocol proposed in this paper, we further elaborate on this proposal below.

Furthermore, lots of researches look for a P2P community in which members behave in a uniform fashion. Thus, abuse and misuse of the system, and the free-riding attack could be combated. In particular, sharing can be encouraged by imposing some form of payment on the downloads (e.g. a proof of computational effort based on solving a puzzle). For instance, recent works have also studied how to use a P2P network to prevent Denial of Service (DoS) attacks on the Internet [6, 8].

Quality of the content is a noteworthy problem for users of a file sharing system. Currently, digital content protection attempts to eliminate suspicions that the content has been modified in a non-authorized way, thus generating mistrust among the community. Several investigations agree with requiring the attention of a human in order to perform a task before granting access or service. This mentioned task has been also recognized as puzzle due to the required effort to solve it, even though this approach affects to the quality of the service (connection availability, bandwidth and storage) provided by a legitimate user. The use of computational puzzles to encourage fair resource allocation has been considered in [5]. Nevertheless, a critical issue is that clients need to trust each other in getting the requested content after they solve the puzzle.

In this paper we explore the design of a secure content distribution protocol for P2P networks, together with a model of content access control by means of cryptographic puzzles. For this, we rely on some results derived from well-studied problems, such as those of reaching consensus in presence of traitors. Content authenticity confirms non-alteration of the content, as well as source identification. As we will see later, this will be implemented through a concatenation of the content's digital signatures among trusted members.

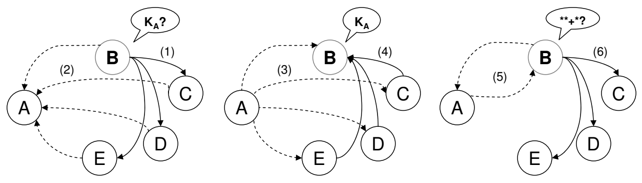

Figure 1. Pathak-Iftode's public key authentication protocol.

The paper is organized as follows. In Section 2, we provide a brief overview of some related work. Section 3 introduces our content authentication protocol. Finally, Section 4 outlines some future research directions and concludes the paper.

2 Related work

Pathak and Iftode [10] apply the ideas presented in the Byzantine Generals problem [7] for public key authentication in pure P2P systems, where generally one cannot assume the existence of a public key infrastructure (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 [4]). 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 in four phases: admission request, challenge response, distributed authentication and Byzantine agreement (see Fig. 1). 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 from a subgroup of its trusted members for verifying its authenticity (Fig. 1(1)). 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 labeled as (2) at Fig. 1. Alice will be able to return the recovered nonce in a signed response message (Fig. 1(3)) if and only if she holds the corresponding private key (K_A^{-1}). Each challenger waits for an application specific time-out, 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 (Fig. 1(4)). If all peers are honest, then there will be a consensus and Bob gets 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 proof request message. Alice must respond with all the challenge messages received and her respective responses ((Fig. 1(5)). 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 (Fig. 1(6)), 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.

2.1 Rational exchange and puzzles

Research on the application of cryptographic puzzles in P2P systems has been mainly oriented to impede DoS attacks and also to provide a solution for the free-riding problem. Puzzles can be used to encourage system users to act in such a way as to increase overall social welfare, either by sharing or rating content. Sharing can be encouraged by imposing a cost on the downloads, but ensuring that those who share more freely do not incur this cost. Thus, users of a system are given an incentive (e.g. *micropayments* [9]) to work together towards a common goal through the introduction of these puzzles.

In essence, the suggested idea is challenging the requester before supplying the required content. A user who do not pass the challenge would be denied access to requested material. "Proof of Effort" [11] and "Proof of Work" [5] have been developed for several purposes, such as timed commitments, time-lock puzzles, cost functions, CPU pricing functions, and moderately-hard memory bound functions [1]. These primitives have been applied as a defense against spam, connection depletion, and attrition attacks, among others.

The main disadvantage related to schemes based on puzzles is that they cannot guarantee a fair result for the party that develop the proof of work. For instance, once the puzzle is resolved and its solution sent to the provider, this is not enforced to behave well and deliver the content. A fair-exchange protocol would resolve this question. However, a fair-exchange protocol cannot be generally implemented in environments such as a pure P2P network, as we assume that TTPs does not exist. It is precisely in this context (i.e. faced with the impossibility of providing true fairness) where notions such as *rationality* become particularly interesting. This concept, widely known to game theorists, was applied to security protocols by Syverson in 1998 [12]. Informally, a rational exchange protocol cannot provide fairness, but it ensures that rational (i.e. self-interested) parties would have no reason to deviate from the protocol, as misbehaving does not result in any benefit. Since rational exchange protocols provide fewer guarantees, one would expect that they also demand fewer requirements, so they can be viewed as a trade-off between complexity and true fairness. In particular, rational exchange protocols do not need a TTP.

3 A P2P Content Authentication Protocol

This section is structured in three parts. First, we introduce some notation and discuss a few working assumptions. Next we motivate the need for a non-trivial content authentication protocol in highly decentralized systems. Finally, we present our scheme based on Byzantine agreement for authentication and cryptographic puzzles for a secure content download.

3.1 Assumptions and Notation

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

1. **Identification** of all participants is required through a unique identifier (pseudonym, IP address, a 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 [10].

Next we summarize the notation used in this paper:

- N is the number of network nodes. Each node is denoted by n_i. Occasionally, specific nodes will be designated by capital letters: A, B, $\ldots$

- 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 secure cryptographic hash function on m.

- $s_{n_i}(x)$ is n_i's signature over x, i.e., $s_{n_i}(x) = enc_{K_{n_i}^{-1}}(h(x))$, whereas $s_{n_i}^{n_j}(x)$ is n_i's signature over x concatenated with n_j's identity, i.e., $s_{n_i}^{n_j}(x) = enc_{K_{n_i}^{-1}}(n_j||h(x))$.

- $\omega(x)$ is a weakly secret bit commitment (WSBC) over x. Basically, it can be viewed as a puzzle whose solution is x. For our purposes, it will take the form of $\mathcal{P}_m^B = w(K_S)$. It denotes a puzzle sent to B, who wants to obtain the key required to decipher m.

3.2 Motivation

One of the most interesting aspects of P2P systems is the possibility of replicating the same content among different nodes. In a sense, replication also guarantees a sort of fault tolerance, since information can be available even if some parts of the network are temporarily disconnected. When properly deployed and managed (e.g. in a collaborative environment), the previous features are highly desirable.

However, it is unrealistic to assume that every integrating node will 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. A malicious party can modify the replica according to several purposes, such as:

- To claim the ownership of the content.

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

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

We can conceive many applications in which it is crucial for a user to be sure about the authenticity of a document. In classic networking paradigms, guarantees of authenticity and integrity can be provided by schemes based on digital signatures. A naive example is that wherein every authenticated user, A, who wishes to offer a content m, provides m together with a content certificate of m. This is simply a signed data structure linking the content (for instance, by using a hash function) with its owner.

The reasons why A cannot sign her own content certificate are straightforward. First, because she can misbehave, offering something different of what she announces. Furthermore, her signature alone does not prevent from manipulation. Suppose that A offers m in the form of a pair $< m, s_A(m) >$. Once B obtains m, she can modify it, obtaining m', and then generate a new signature over the altered content. Moreover, even if B does not modify m, she can just remove $s_A(m)$ and add her own signature. As a result, several –and probably different– copies of m claimed by various parties may be circulating through the network.

To solve this problem, the community must rely in the existence of an external TTP, which is is in charge of signing every content. However, this assumption is an unrealistic supposition in many P2P systems, especially in mobile environments (e.g. P2P applications over wireless ad-hoc networks), in which one cannot assume that a proper infrastructure will be available. In these scenarios, a flexible and autonomous scheme is required, in which nodes can securely manage data without relying in external elements.

3.3 Proposed Scheme

Our proposal consists of three main stages: a signature generation subprotocol, a rational content access subprotocol, and a signature verification subprotocol. Fig. 2 summarizes the scheme.

3.3.1 Signature Generation Subprotocol (SGS)

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 $\mathcal{S}$, 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 can depends, for example, on their availability, level of trust, etc.

Next, A sends to each member $n_i \in \mathcal{S}$ a message containing the content m, and A's signature over it:

$$A \to n_i \in \mathcal{S} : m, s_A(m)$$

Each $n_i \in \mathcal{S}$ should verify the authenticity and integrity of m, as well as to authenticate the originator. A crucial issue is that a node n_i must not sign twice the same content coming from different originators. For this, n_i maintains a table of signatures, denoted T_i, in which each signed message is stored in a new entry. Table T_i has the following structure:

$$\langle ts, A, h(m), s_A(m) \rangle$$

where ts is a timestamp, A is the node who claims to be owner of m, $h(m)$ is a hash of m, and $s_A(m)$ is the signature provided by A.

Once n_i has agreed to collaborate with A, she signs m linked to A's identity and sends the signature to A:

$$n_i \in \mathcal{S} \to A : s^A_{n_i}(m)$$

A must check every signature received using, if necessary, Pathak and Iftode's protocol [10]. Then, she publishes the following records (note that m is not published):

$$s^A_{n_1}(m)||s^A_{n_2}(m)|| \cdots ||s^A_{n_k}(m)||enc_{K_A}\big(h(m)\big)$$
$$n_1, n_2, \ldots, n_k, A$$

- **`Signature Generation Subprotocol`**
  1. A `generates` $h(m)$ `and signs it:` $s_A(m)$
  2. `For` $i = 1$ `to` k
     (a) A `sends` $(m, s_A(m))$ `to` n_i
     (b) n_i `verifies the correctness of` $h(m)$ `and checks` T_i
     (c) n_i `sends its signature` $s^A_{n_i}(m)$ `to` A
     (d) n_i `adds the new entry to her table of signatures` T_i
  3. A `verifies the correctness of the signatures, encrypts` $h(m)$`, and publishes:`
     $$s^A_{n_1}(m)||s^A_{n_2}(m)|| \cdots ||s^A_{n_k}(m)||enc_{K_A}\big(h(m)\big)$$
     `together with the identities of nodes:`
     $$n_1, n_2, \ldots n_k, A$$
- **`Rational Content Access Subprotocol`**

$$\begin{aligned} B \to A\text{:} \quad & m_1 = enc_{K_A}\big(h(m)\big), \sigma_1 \\ A \to B\text{:} \quad & m_2 = enc_{K_S}(m), enc_{K_B}(\mathcal{P}^B_m), \sigma_2 \\ \text{where} \quad & \sigma_1 = sig_B\big(enc_{K_A}\big(h(m)\big)\big) \\ \text{and} \quad & \sigma_2 = sig_A\big(enc_{K_S}(m), enc_{K_B}(\mathcal{P}^B_m)\big) \end{aligned}$$

Figure 2. Proposed scheme.

3.3.2 Rational Content Access Subprotocol (RCAS)

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. Query result should return at least the content descriptor and the list of identities of the source nodes. Together with each query result, B obtains the list of signers and their signatures associated with the content, as explained in the previous section.

Before verifying the signatures, B must select a source A according to some criterion, e.g. trust on some of the signers, or simply at random. B must initiate the content access subprotocol once A is chosen. Now, B signs the last part of the content's information published, $sig_B\big(enc_{K_A}\big(h(m)\big)\big)$, and sends it to A. Then, A can check B's identity and computes a session key K_S with which encrypt m. K_S is also sent to B by means of a puzzle related to m and B, $\mathcal{P}^B_m$.

$$\mathcal{P}^B_m = \omega(K_S)$$

Therefore, B will have to solve $\mathcal{P}^B_m$ for getting m.

3.3.3 Signatures Verification Subprotocol (SVS)

Once B gets the desired content, she must verify the correctness either of some or all of the concatenated signatures and their identities. 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 on the weighted values given by a Trust Management System (TMS). Since B has guarantees of the correct performance of the SGS, she can believe in content authenticity. In particular, robustness of hash functions assures content integrity, as B can always verify that content has not been modified. On the other hand, digital signatures ensure node authentication, thus allowing B to verify who has created the signature. Moreover, the number k of signers is global and known by client application.

3.4 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 must generate again the last item of the published information about m, i.e. $h(m)$ encrypted with B's public key. 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.

Our proposal alone cannot prevent m from being modified by B once she has got access to it. This is a different problem, which must be addressed by another means (e.g. watermarking, fingerprinting, etc). However, if B publishes a modified replica, any requester will be able to detect it just by checking the associated signatures.

4 Conclusions and research directions

In this paper, we have presented a content authentication protocol designed for pure P2P systems, which are mainly characterized by the node transience and the lack of a centralized authority. The proposed scheme allows for a secure content replication among peers, maintaining integrity control over the published contents. Furthermore, our proposal includes a cryptographically-based countermeasure against P2P non-authorized content alteration, in which proofs of computational effort, may serve as a rational content access control. Among the possible ways in which rationality can be ensured, we have considered puzzles as incentives for users to behave in a uniform fashion.

Our motivational 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 the source has lost control over it. Currently, we are implementing this scheme in J2EE to evaluate performance and robustness in several attack scenarios, such as DoS, cheating, and free-riding.

References

[1] M. Abadi, M. Burrows, M. Manasse, and T. Wobber. Moderately hard, memory-bound functions. *ACM Transactions on Internet Technology*, 5(2):299–327, May 2005.

[2] E. Damiani, S. D. Capitani, S. Paraboschi, P. Samarati, and F. Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 207–216, Washington,USA, November 2002. ACM.

[3] N. Daswani and H. Garcia-Molina. Pong-cache poisoning in guess. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 98–109, Washington, USA, October 2004. ACM.

[4] J. Douceur. The sybil attack. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems*, pages 251–260, Cambridge, USA, March 2002.

[5] 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.

[6] 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.

[7] L. Lamport, R. Shostak, and M. Pease. The byzantine general problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[8] 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.

[9] D. Mankins, R. Krishnan, C. Boyd, J. Zao, and M. Frentz. Mitigating distributed denial of service attacks with dynamic resource pricing. In *Proceedings of the 17th Annual conference on Computer Security Applications*, June 2001.

[10] 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.

[11] D. Rosenthal, M. Roussopoulos, P. Maniatis, and M. Baker. Economic measures to resist attacks on a peer-to-peer network. In *Proceedings of the Workshop on Economics of Peer-to-Peer Systems*, Berkeley, USA, June 2003.

[12] P. Syverson. Weakly secret bit commitment: Applications to lotteries and fair exchange. In *Proceedings of the 11th IEEE Computer Security Foundations Workshop*, pages 2–13, 1998.