A P2P File-Sharing Protocol based on Cryptographic Puzzles

Esther Palomar, Juan M. Estevez-Tapiador,
Julio C. Hernandez-Castro, and Arturo Ribagorda

Computer Science Department – Carlos III University of Madrid
Avda. Universidad 30 – 28911, Leganes, Madrid
{epalomar, jestevez, jcesar, arturo}@inf.uc3m.es

Abstract. Pure peer-to-peer (P2P) networks present a number of challenges from a security standpoint. Most of the difficulties found to apply classic security solutions are related to the inherent lack of central authorities, and new proposal have to deal with avoiding such a centralization by exploring alternative paradigms, which often require collaboration among peers. In particular, access control in current P2P file sharing systems is chaotic, or even nonexistent. This fact motivates us to propose a protocol which employs proof-of-work to evaluate, to some extent, requesters' privileges gained by means of parameterizable cryptographic puzzles. Roughly, in our proposal contents are always stored and transmitted encrypted. Each requester is provided with a trapdoor (a puzzle) whose difficulty varies according to his privileges. The successful resolution of this puzzle provides him with the needed key to access the content. Experimental results are reported to illustrate the computational consumption and the time spent by the protocol according to several cryptographic algorithms.

1 Introduction

P2P file sharing systems are widely recognized for theirs valuable capabilities and underlying advantages. In pure P2P systems (e.g. mounted on top of a mobile ad-hoc network) one cannot generally assume the existence of any fixed infrastructure, such as a public key infrastructure (PKI). This fact has immediate consequences, particularly from a security point of view. When a content is replicated through various locations, the owner loses control over the replicas. Any file-sharing protocol should ensure that a content's provider (be its owner or not) is authorized to render that particular service, and that the content has not been modified in a non-authorized way. At the same time, the provider should verify that the requester has enough privileges to access the content. The absence of these elemental security services is certainly a serious concern for critical applications, while for others (e.g. file sharing in Internet) it may be simply a source of mistrust among the community members (existence of fake contents, disputes for the ownership of a content, etc.)

In a fully distributed system (e.g. a pure P2P network), the provision of security services such as those mentioned above become quite difficult if public-key

cryptography cannot be used –at least, in the way it is employed in classic distributed systems. Nearly all the solutions proposed so far are one way or another based on the idea of replacing the whole PKI by a collaboration scheme. Put simply, a subgroup of peers must cooperate to perform tasks such as generating or verifying a digital signature. Threshold cryptography has been repeatedly pointed out as an appropriate technique to achieve these purposes, even though its computational cost can disable its application for restricted devices (e.g. mobile scenarios.)

A major problem, however, is to guarantee that all the community members will collaborate to jointly carry out a process. Some studies have pointed out the benefits of employing incentive-based mechanisms as a means to foster cooperation among peers. Even though these incentives rarely can substitute the strength provided by a cryptographic primitive, they can be still very useful as a trade-off for non-critical applications. For instance, incentives have demonstrated its applicability to combat free-riders (users that get contents but share nothing.)

In this paper, we present an authorization protocol which provides access control to contents anonymously. Our approach relies on an interactive challenge-response scheme aimed at providing local access control on each individual node's resources. In essence, the suggested idea is challenging the requester before supplying the required content. A user not passing the challenge will have to face an almost impossible task for getting the requested material. The protocol is anonymous in the sense that providers and requesters do not mind who the other exactly is, whenever the latter responds to the challenges with the correct answer. The challenge is basically a puzzle whose difficulty may depend on some kind of reputation history (much in the way incentive techniques do), thus creating the possibility that a previously well-behaved peer may be rewarded.

The rest of this paper is organized as follows. In Section 2, we give a brief description of some proof of work protocols and its application in P2P networks. Section 3 describes our proposal, while Section 4 is devoted to present some analysis concerning the efficiency and security of our proposal through a number of scenarios. Finally, Section 5 concludes the paper and discusses some open issues and directions for future work.

2 Background and related work

Proof-of-effort and proof-of-work techniques [1] have been developed for several purposes, such as timed commitments, time-lock puzzles, cost functions, CPU pricing functions, and moderately-hard memory bound functions (MBF) [2]. These primitives have been applied as a defense against spam, connection depletion, and attrition attacks, among others. The idea of using cryptographic puzzles to increase the cost of sending email and make sending spam unprofitable has been extended to P2P networks in order to encourage fair resource allocation [3]. On the other hand, research on the application of cryptographic proof of work protocols in P2P systems has been mainly oriented to impede Denial of

Service (DoS) attacks, and also to provide a solution for the free-riding problem [4–6] and access control [7]. Basically, peers can stem the flood of requests by demanding that each request be accompanied by a proof of computational effort. Some forms of cryptographic puzzles can also be used to avoid Sybil attacks where attackers with a large amount of computational resources can get a huge range of node IDs [8].

A recent work [9] applies the ideas presented in the Byzantine Generals problem [10] for public key authentication in pure P2P systems, by means of proof of possession. They postulate that a correct authentication depends on an honest majority of a particular subgroup of the peers' community, labeled "trusted group". Thus, honest members from trusted groups are used to provide a functionality similar to that of a CA (certification authority) through a consensus and challenge-response procedure.

Some works have focused on quantifying the cost/benefit tradeoffs that will lead nodes to share theirs resources. In this context, node participation in ad hoc and P2P networks is recently addressed in [11] adopting a game theoretic approach. From the results presented, authors conclude that even if nodes perceive a cost in sharing their resources, this may induce node participation. This happens as nodes recognize that refusing to participate will result in similar behavior by others, which consequently would compromise future transactions, and obviously the viability of the network as a whole. However, advanced security issues such as the zero-cost-identity problem and collusion between free-riders must be careful addressed [12].

A significant amount of research has focused on developing micropayments protocols in P2P systems [13, 14]. Specifically, commercial file sharing systems have addressed different techniques to face up free-riding and authors' copyright problems. In particular, this research area tends to meet fairness, assuring that, on one hand, authors are guaranteed to be paid, and on the other hand users are discouraged to behave as freeloaders because they are refunded for each upload. Thus, system users are given an incentive to work together towards a common goal through the introduction of these puzzles. Most of the presented proposals link authorization with a "grantor" (a TTP), who produces certificates binding owners to contents, owners to requesters, and also owners to theirs providers. All verification and accounting loads fall on a unique entity, what introduces several performance and delegation inconveniences.

The main disadvantage related to schemes based on puzzles is that they cannot guarantee a fair result for the party that carry out the proof of work. Once the puzzle is solved and its solution sent to the challenger, this is not enforced to behave well and deliver the reward. Even though fair-exchange protocols address such they cannot be generally implemented in environments such as a pure P2P network, where TTPs do not exist. It is precisely in this context where notions such as *rationality* become particularly interesting. This concept, widely known to game theorists, was applied to security protocols by Syverson in 1998 [15]. 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. In particular, rational exchange protocols do not need a TTP.

3 Granting access based on proof of work

In this section we first introduce some notation and discuss a few working assumptions. Next we briefly state the need for a non-trivial challenge-response mechanism to provide access for file sharing in highly decentralized systems. Finally, we present our proposal.

3.1 Requirements and notation

We assume the following requirements:

1. We take into account the main characteristics of P2P systems, such as:
   - Collaboration-based applications have to deal with the social and rational dilemma; asymmetry exists on peer's interests.
   - Anonymity makes impossible for peers to determine others' identities.
   - Highly transient population and the lack of fixed infrastructure makes difficult and inefficient some tasks such as authentication and accounting.
2. Concerning identification, nodes may use pseudonyms and alias to identify themselves. A user's public key may be associated with a nickname like some web-of-trust–based systems [16]. Moreover, a unique descriptor, which is also used for searching the content, is associated with the content.
3. Digital signatures cannot be forged unless the attacker get access to private keys.
4. Anyone can verify the authenticity of a node's signature by applying the Byzantine fault tolerant public key authentication protocol presented in [9].
5. Transactions are assured. The absence of a message can be detected. This can be provided by using a scheme based on timeouts.

Now, we summarize the notation that will be used throughout the 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_{n_i} and $K_{n_i}^{-1}$, respectively. In turn, $enc_{K_{n_i}}(x)$ represents the (asymmetric) encryption of message x using K_{n_i} as key.
- $E_{K_S}(x)$ denotes the symmetric encryption of x using session key K_S.
- m denotes the content that nodes wish to publish, and m_i its identifier.
- $h(x)$ represents a hash function on x.
- $s_{n_i}(x)$ 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.:

$$s_{n_i}^{n_j}(x) = enc_{K_{n_i}^{-1}}(n_i||h(x))$$

– $\mathcal{P}_m^{n_i}(d, x, n_j)$ represent a $d-$difficult cryptographic puzzle ($d \in [0, \ldots, d_{max}]$) whose solution is x, issued by n_i to n_j, and related to the content m.
– $\omega(x)$ is a weakly secret bit commitment (WSBC) function over x. Basically, it can be viewed as a puzzle whose solution is x. For our purposes, it will take the form of $w(K_S)$. It denotes a weakly puzzle to obtain the key required to decipher the corresponding m. See Section 3.2 for further details.

3.2 Cryptographic puzzles

There exists many ways of constructing a function such that its computation is quite difficult unless a party posses 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 hard 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. Another 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 $\mathcal{P}_m^{n_i}(d, x, n_j)$.

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

3.3 Content sharing

The entire scheme is illustrated in Fig. 1. In step one, before content dissemination, owners always performs the following stages:

– First, the owner A of the content m will generate a non-repudiable and unforgeable evidence ensuring the authenticity of m. For this, A first selects a subgroup $n_i \in \mathcal{S}$, with $|\mathcal{S}| = k$, among her group of trusted members, and sends to them a message containing the content m, and A's signature over it:

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

– Each participant should verify the authenticity and integrity of m, as well as to authenticate the originator. Once n_i has agreed to collaborate with A, she signs m linked to A's identity and sends the signature to A:

$$n_i \in \mathcal{S} \rightarrow A : s_{n_i}^A(m)$$

- `Content sharing`

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 computes` K_S `to encrypt` m.
4. A `publishes:`

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

- `Asking for a trapdoor`

1. $B \rightarrow A$: $m_1 = enc_{K_A}\big(h(m)\big), \sigma_1$
2. $A \rightarrow B$: $m_2 = enc_{K_B}(\varsigma_j), \sigma_2$
3. $B \rightarrow A$: $m_3 = enc_{K_A}(\tau_j), \sigma_3$
4. $A \rightarrow B$: $m_4 = enc_{K_B}(\omega_m), \sigma_4$

$$\begin{aligned} \text{where } \sigma_1 &= s_B\big(enc_{K_A}\big(h(m)\big)\big) \\ \sigma_2 &= s_A\big(enc_{K_B}(\varsigma_j)\big) \\ \sigma_3 &= s_B\big(enc_{K_A}(\tau_j)\big) \\ \text{and } \sigma_4 &= s_A\big(enc_{K_B}(\omega_m)\big) \end{aligned}$$

Fig. 1. Proposed scheme.

– A must check every signature received using, if necessary, Pathak and Iftode protocol [9]. Then, a session key K_S is computed, securely kept, and used to symmetrically encrypt m: $E_{K_S}(m)$. This represents the maximum costly puzzle.
– Moreover, A encrypts $h(m)$ using her own public key. This will be useful when honest requesters demand a trapdoor.
– Finally, A publishes the following record (note that m is always transmitted encrypted):

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

together with m_i, and the identities of nodes:

$$n_1, n_2, \ldots n_k, A$$

Now, let B be the requester who desires m. We can assume that, at this time, B is already engaged in a searching process which typically leads to a list of sources that keep a replica of the desired content. Query results should return at least the content descriptor, the list of identities of the source nodes, and sources' published information, as explained above.

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. Once the content is downloaded, B has to decrypt $E_{K_S}(m)$ for getting m. Thus, B must choose between the following two options:

- Ask A for a trapdoor (see Section 3.4).
- Try to solve the maximum hardness puzzle included in the search answer.

Once B decrypts the content, he should verify the correctness of some or all of the concatenated signatures and their identities.

3.4 Distribution and computation of puzzles

Before B can get a trapdoor, he has to solve a challenge issued by A. For this, B contacts A using the last part of the content's information published, signs it, and sends it to A: $s_B\big(enc_{K_A}\big(h(m)\big)\big)$ (see Fig. 1–bottom). Then, A can check B's identity and the desired content's hash.

Now, A performs a sequence of verifications on her local past transactions history, T_A. This table has the following structure:

$$\langle n_j, h(m), \varsigma, \tau, t_i, L_{n_j} \rangle$$

where ς is the challenge sent to n_j, τ his response, time t_i the corresponding timestamp of the transaction, and L_{n_j} represents the security label n_j has got at that moment. This clearance can depend on different parameters (time in responding to the challenge, its difficulty, etc.) and are issued discretionally by providers. For instance, a requester's security level may be increased according to the number of successful interactions with the provider.

Concerning types of challenges, we use a moderately-hard memory-bound (MHMB) function as proposed by Abadi et al. [2]. Roughly, $F()$ is a MHMB function whose domain and range are integers in $0 \ldots (2^n - 1)$, and we suppose that its inverse $F^{-1}()$ cannot be evaluated in less time than a memory access. The key idea is that F() is chosen so that the verification by the challenger is fast, and so that the computation by the requester is fairly slow. Then, A returns a fresh challenge ς to B. The latter should compute $F(\varsigma)$ and returns τ to A. Finally, A verifies that what it receives is a correct response to ς, and, if so, creates a trapdoor, $\omega(K_S)$, for getting m.

The trapdoor is also sent to B and associated with m, A and B, as follows:

$$\mathcal{P}_m^A(l_{bits}, K_S, B) = \omega(K_S)$$

where the number l of key bits provided depends on the new L_B reached.

B must explore, on average, 2^{n-l} different keys –n being the total number of bits of K_S–, until m is correctly deciphered. Note that A may supply many different trapdoors for the same m choosing randomly l bits of K_S. Additionally, the system may maintain the requester's effort feedback where honest requesters can announce the cost of theirs probes, and will be also stored in the corresponding T_A's entry.

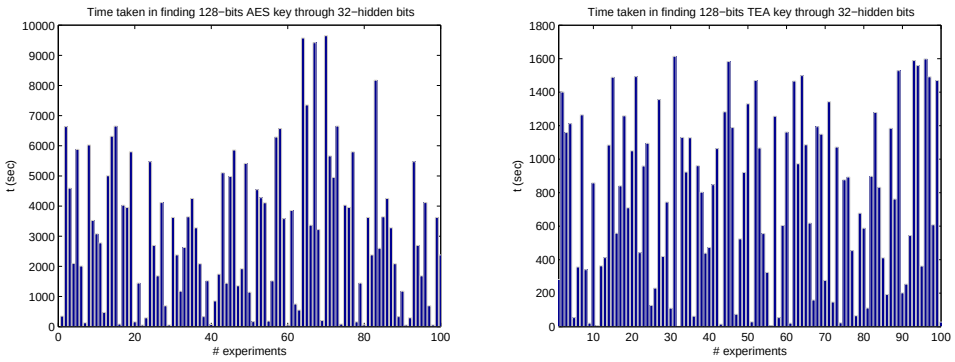

Fig. 2. AES (left) and TEA (right) computational consumption in seconds to find the corresponding K_S through a 32-bits trapdoor in a sample of 100 experiments.

4 Discussion

This section presents an efficiency analysis considering the computational (especially time) effort required by honest nodes to follow the scheme, as well as some security considerations and vulnerabilities.

4.1 Simulation Framework

We have considered two block ciphers as the basis for cryptographic puzzles, AES-128 and TEA [17, 18]. Both were coded in C, compiled with Microsoft Visual C++, and ran on a AMD ATHLON(tm)2600 2.09GHz processor, 1GB RAM under Windows XP SP2.

The factor responsible for the complexity is the cost of executing several decryptions, testing each candidate key. We have carried out 1000 experiments for different values of l-bits $(0 \dots 2^{32})$, and also randomly varying the challenges and key used. We used the Mersenne Twister algorithm [19] for generating uniform pseudorandom numbers.

It is interesting to examine to what extent the usage of this kind of effort-aware access control can be applicable and reliable in real networks. First, we have considered that more than 32 hidden bits would result in a impracticable number of potential keys to test in a common platform.

The average results are shown in Table 1. For each case, as the l threshold increases, the number of candidate keys obviously decreases, so the exploration time too. Fig. 2 presents the exploration effort using both algorithms and trapdoors with 32 hidden bits, in a sample of 100 experiments. In this case, AES requires 90 minutes of computational effort, while TEA takes 27 minutes, on average. Therefore, we can consider them good examples of puzzles for our work-aware access control model. Fig. 3 compares both schemes in terms of different amount of supplied l bits.

ω bits	l bits	Time required on average (sec)	
		AES	TEA
32	96	5495	761
28	100	544	47
24	104	15	0.22
20	108	0.01	0.01

Table 1. Computational effort spent on average (in 1000 experiments) by each algorithm regarding the amount of given bits.

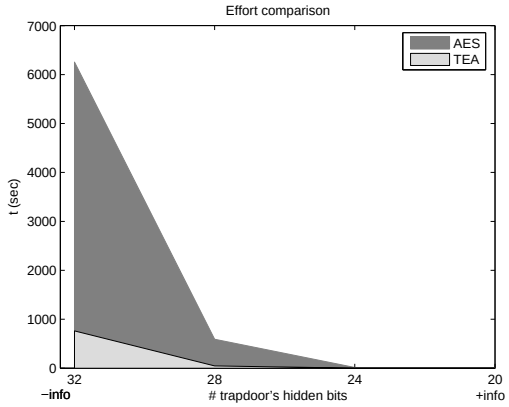

Fig. 3. Comparison of both algorithms' computational effort spent on average (in 1000 experiments) regarding the amount of given information of the encryption key.

The complexity increases when the given l key bits are not organized in the same key blocks, but randomly. Moreover, decryption would be difficult if the key length is also hidden, e.g. using AES 256-bit key, 256-bit block mode.

4.2 Security considerations and vulnerabilities

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

Cheating and man-in-the-middle. Suppose an eavesdropping peer can observe the messages tagged as $m_{1...4}$ at Fig. 1. In theory, she is looking for the proper K_S for getting m. This message reveals nothing about ω, since they are all correctly encrypted using the recipient's public key. On the other hand, dishonest nodes may try to use others' computational resources for solving hers challenges and puzzles, like a type of free-riding situation. Basically, a malicious node C, decrypts and uses the challenge received from one of hers providers (say

A) and sends it to challenge one of hers requesters, B. Since B is honest, he solves and sends the puzzle solution to C who decrypts it and encrypts it again but using K_A. A will receive the correct answer sent by C but solved by B. This is related to the so-called *grandmaster postal-chess problem* [20]. For this reason, puzzles must been associated with each requester, and with the content as well. Consequently, it will not be able to have any "zombie" for delegating puzzle computation either.

Dishonest providers. A real problem occurs when the provider refuses to send the appropriate trapdoor after receiving the correct puzzle response. Fair exchange and zero-knowledge protocols address this kind of misbehavior [21]. Due to space limitations, we can not give further details, although we stress that message m_1 could be used as a proof of A's misbehavior. In particular, this message can be used as a non-repudiation evidence by the requester.

Impersonation. A major security concern of challenge-response-based schemes is adversaries who eavesdrop and subsequent attempt to impersonate the role of an honest peer. Impersonation may be trivial if an adversary is able to discover a nodes long-term (secret or private) keying material, using for example a chosen-text attack. Briefly, suppose that authentication here consists of A challenging B with a random number r_A, RSA-encrypted under Bs public key, and B is required to respond with the decrypted r_A. If A challenges B with $r_A = h(x)$, Bs response to this authentication request may (unwittingly) provide to A its RSA signature on the hash value of the (unknown to B) message x. This may be possible in protocols which are not zero-knowledge, because the claimant uses its private key to compute its response, and thus a response may reveal partial information. Other active attacks may involve the adversary itself initiating one or more new protocol runs, and creating, injecting, or otherwise altering new or previous messages [22]. Witness-indistinguishability and proof-of-knowledge techniques [23] deal with this issues. Our model should deal with them through embedding in each challenge response a self-chosen random number, combining use of random numbers with short response time-outs and using timestamps.

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

Off-line periods and non-collaboration. Considering off-line periods, although our scheme provides efficient and secure content replication, but is le-

nient toward non-collaborative and faulty peers: a proportion of signers must cooperate in content integrity process at each time, on the other hand providers and requesters exchange a few messages among each others to conveniently grant access. However, rational parties would have no reason to deviate from the protocol.

5 Conclusions and Future Work

In this paper, we have proposed an scheme based on the use of cryptographically-based countermeasures against P2P non-authorized content access, 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 computational puzzles as incentives for users to behave in a uniform fashion, and discourage free riders.

Experiments show us signs of the reliability of the proposal according to users' computational resources and interests. We, as many authors, try to probe that peers tend to collaborate even if they must spend some resources, and playing down the importance of indirect trust. Therefore file sharing can be encouraged by imposing a cost on the downloads, where each transaction implies to be worthy of access each time.

Finally, the application of threshold cryptography and consensus-based techniques for providing authorization in P2P networks are also interesting research lines that will be tackled in future works. New experiments are being simulated using different cryptographic algorithms such as hash functions and stream ciphers.

References

1. Rosenthal, D., Roussopoulos, M., Maniatis, P., Baker, M.: 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 (2003)
2. Abadi, M., Burrows, M., Manasse, M., Wobber, T.: Moderately hard, memory-bound functions. **5** (2005) 299–327
3. Dwork, C., Naor, M.: Pricing via processing or combatting junk mail. In: Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology. Volume 740. (1992) 139–147
4. Juels, A., Brainard, J.: Client puzzles: A cryptographic defense against connection depletion attacks. In: Proceedings of the Networks and Distributed Security Systems, California, USA (1999) 151–165
5. Maniatis, P., Giuli, T., Roussopoulos, M., Rosenthal, D., Baker, M.: Impeding attrition attacks in p2p systems. In: Proceedings of the 11th ACM SIGOPS European Workshop, Leuven, Belgium, ACM (2004)
6. Zhou, F., Zhuang, L., Zhao, B., Huang, L., Joseph, A., J.Kubiatowicz: Approximate object location and spam filtering on peer-to-peer systems. In: Proceedings of the ACM International Middleware Conference, Rio de Janeiro, Brazil, ACM (2003) 1–20

7. Palomar, E., Estevez-Tapiador, J., Hernandez-Castro, J., Ribagorda, A.: Certificate-based access control in pure p2p networks. In: Proceedings of the 6th International Conference on Peer-to-Peer Computing, Cambridge, UK, IEEE (2006) 177–184
8. Borisov, N.: Computational puzzles as sybil defenses. In: Proceedings of the 6th Int. Conf. on Peer-to-Peer Computing, IEEE (2006) 171–176
9. Pathak, V., Iftode, L.: Byzantine fault tolerant public key authentication in peer-to-peer systems. Computer Networks (2006)
10. Lamport, L., Shostak, R., Pease, M.: The byzantine general problem. ACM Transactions on Programming Languages and Systems **4** (1982) 382–401
11. DaSilva, L., Srivastava, V.: Node participation in ad hoc and peer-to-peer networks: A game-theoretic formulation. In: Proceedings of the Wireless and Communications and Networking Conference, New Orleans, USA, IEEE Computer Society (2005)
12. Banerjee, D., Saha, S., Sen, S., Dasgupta, P.: Reciprocal resource sharing in p2p environments. In: Proceedings of the 4th Int. joint conference on Autonomous agents and multiagent systems, The Netherlands, ACM Press (2005) 853–859
13. Anceaume, E., Gradinariu, M., Ravoaja, A.: Incentives for fair resource sharing. In: Proceedings of the 5th IEEE International Conference on Peer-to-Peer Computing, Konstanz,Germany, IEEE (2005) 253–260
14. Catalano, D., Ruffo, G.: A fair micro-payment scheme for profit sharing in a p2p networks. In: Proceedings of the Int. Workshop on Hot Topics in Peer-to-Peer Systems, Washington, USA, IEEE Computer Society (2004) 32–39
15. Syverson, P.: Weakly secret bit commitment: Applications to lotteries and fair exchange. In: Proceedings of the 11th IEEE Computer Security Foundations Workshop. (1998) 2–13
16. Anonymous Communication With Waste. http://waste.sourceforge.net.
17. Daemen, J., Rijmen, V.: The Design of Rijndael: AESThe Advanced Encryption Standard. Springer-Verlag, Berlin Germany (2002)
18. Russell, M.: Tinyness: An overview of tea and related ciphers (2004) http://www-users.cs.york.ac.uk/ matthew/TEA/TEA.html.
19. Matsumoto, M., Nishimura, T.: Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Trans. Model. Comput. Simul. **8** (1998) 3–30
20. Menezes, A., Oorschot, P.V., Vanstone, S.: Capter 10. Identification and Entity Authentication. In: Handbook of Applied Cryptography. CRC Press (1996) 385–424
21. Alcaide, A., Estevez-Tapiador, J., Hernandez-Castro, J., Ribagorda, A.: An extended model of rational exchange based on dynamic games of imperfect information. In: Proceedings of the Int. Conf. on Emerging Trends in Inf. and Comm. Security, Germany (2006) 396–408
22. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press (2001)
23. Bellare, M., Palacio, A.: Gq and schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In: Proceedings of the 22nd Annual Int. Cryptology Conf. on Advances in Cryptology, London, UK, Springer-Verlag (2002) 162–177