Effects of Cooperation–based Peer–to–Peer Authentication on System Performance

Esther Palomar, Arturo Ribagorda
Department of Computer Science
University Carlos III of Madrid, Spain
Email: {epalomar,arturo}@inf.uc3m.es

Juan E. Tapiador
Department of Computer Science
University of York, UK
Email: jet@cs.york.ac.uk

Julio Hernandez-Castro
School of Computing
University of Portsmouth, UK
Email: Julio.Hernandez-Castro@port.ac.uk

***Abstract*—Fully distributed file sharing systems, where participants share content and collaborate in a Peer–to–Peer (P2P) manner, have to deal with a highly dynamic environment to ensure that even when some nodes are unavailable, others can still perform the task through the coalition of cooperating parties. Therefore, cooperation–based services have been extensively deployed over the majority of P2P networks. This leads, however, to significant challenges for network services and data management, especially facilitating security properties. In a previous work, we presented a protocol for content authentication based on attribute certificates that does not rely on the existence of a public key infrastructure. Instead, peers cooperate to guarantee data integrity. In this context, it is reasonable to assume that peers would cooperate aimed at maximizing the expected benefit, but, generally, efficiency performance pays dearly for this. In this paper, we present an experimental evaluation of our analytical model in order to validate its dependability, and also discussing its tolerance under failures.**

I. Introduction

The lack of authentication of the content shared by a P2P community may motivate the service offered by such a file sharing system is not totally convincing just yet. The provision of security services, e.g. data integrity, has proven to be quite difficult in most existing P2P file sharing systems due to the absence of a central authority or trusted third party (TTP). Unfortunately, peers can then publish fake or junk files with the names or keywords of some popular files, causing normal users to frequently download wrong files. This quickly makes peers lose interest in that community [1].

Solutions to achieve content authentication, as many other network security services offered today, rely on different cryptographical techniques. Most of the difficulties found to apply classic security solutions (e.g. authentication and authorization services) are just related to the impossibility of deploying a public key infrastructure (PKI) in these environments, and new proposals have to deal with avoiding such a centralization by exploring alternative paradigms, which often require cooperation among peers. For example, Merkle trees [2] have been applied to efficiently build secure authentication and signature schemes from hash functions [3] in DHT-based systems. Time and space complexities have been frequent problems to faced with, closely followed by updating procedure (a modification of one bit in the content leads to very different hashes), mistrusted storage and its cost, and problems of the efficiency when the number of documents is not close to a power of 2. Similarly, threshold cryptography approaches let peers themselves to locally handle main procedures without any dealer [4]. However, the establishment of the threshold t is usually a main matter. With a fixed threshold policy, an adversary may try to manage and permanently compromise t group members in order to expose the secret. In such cases, the reduction of value t involves associated problems such as when a large number of members leave the group, resulting in a new group size, probably less than t.

In this paper, we analyze the impact of using a certificate-based content authentication protocol which firmly depends on the collaboration of the participants. Our goal is to experimentally quantify its confidence in terms of its performance costs.

The rest of this article is organized as follows. We first describes the content authentication protocol in Section II and subsequently, formally evaluate the scheme extending the discussion to the main performance parameters in Section III. Finally, Section IV concludes the paper and outlines some future research directions.

II. Protocol Description

The basic idea is that contents will be associated to a digital certificate ensuring properties such as integrity and authenticity, much in the way an X.509 public key certificate can be used to ensure these properties for a public key. According to RFC 3281 [5], the attributes are digitally signed and the certificate issued by an attribute authority –an entity that pure P2P networks do not have. Instead, anyone will be able to verify the authenticity of a node's signature by applying the Byzantine fault tolerant public key authentication protocol presented in [6]. This scheme adopted in our model depends on an honest majority of a particular subgroup of the peers' community, labeled "trusted group". Concretely, honest members from trusted groups are used to provide a functionality similar to that of the PKI through a consensus procedure. Obviously, this process leads to some performance drawbacks that we must analyze.

First, the details of a content certificate and a full description of the scheme are provided below.

Content certificate $\mathbf{C_m}$

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

Fig. 1. Content certificate.

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

TABLE I
CERTIFICATE GENERATION: SIGNERS' DELIVERABLES.

A. Content Certificate Generation

Let A be the legitime owner of a given content m. After A joins the system and shows interest in distributing m, she must first produce content certificate C_m (see Figure 1 for details), as follows. A selects an appropriate value for the number k of cooperative nodes. These k nodes are also called *signers*. Ordered list of signers (OLS) contains their identities, denoted by $A, n_1, \ldots, n_k$. The way in which this selection is carried out may consider different criteria, e.g. using information about past transactions to make decisions regarding current interactions, or the availability level. For this purpose, we may consider different values for k. We further elaborate on this concern in Section III.

Finally, before C_m can be used to ensure content authenticity and integrity, it must be progressively signed by the nodes included in the OLS. At each stage, the next node in the OLS adds its signature to the previous ones. Due to the structure of the chain of signatures, this task cannot be carried out in parallel, i.e. OLS signatures have to be nested. We will denote $C_0, C_1, \ldots, C_k = C_m$ the successive versions of the certificate as it passes through the list of nodes, i.e. A generates C_0 by providing the first signature and passes it to the next node in the OLS. The chained signature is defined as follows:

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

Signers have to perform the following *local verifications* in order to ensure that they are not being cheated on, as follows. Each signer, $n_i \in$ OLS, should verify the correctness of the received certificate C_{i-1}. This includes: (i) Obtaining the owner's public key, and authenticating previous signers as well, (ii) Computing $h(m)$ and comparing it with the value contained in the received certificate, and (iii) Checking its history of signed contents, and verify that no entries exist corresponding to the same content. In addition, each signer should verify 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. Finally, if previous verifications succeed, the node will add its signature to C_{i-1}, thus creating C_i. Summarizing, A publishes the content m, along with the content certificate: (m, C_m). For the sake of illustration, we include the following Table I which summarizes the content certificate signing process.

B. Guarantees for Content Authenticity

Let B be the requester who wishes to access m. We can assume that, at this time, B has already completed a searching process and obtained a list of sources that keep a replica of the desired content. A query result should return the content's descriptor, which could be necessary to rank the relevance of the resulted content to the query, as well as the list of identities of the source nodes. Together with each query result, B obtains sources' published information, C_m with all the information associated with the content, and m, as explained previously. The correctness of the certificate should be verified to ensure the authenticity and integrity of m, so B must verify the correctness either of some or all of the chained signatures and their identities as well. For this purpose, B performs three steps similarly to way it is done in previous stage.

III. EXPERIMENTAL EVALUATION

For our experimental evaluation we have implemented the content authentication protocol on a working prototype with the main functionalities of file sharing to perform simulations. Results were obtained with an AMD ATHLON(tm)2600 2.09GHz processor, with 1GB RAM under Windows XP SP2, and Java Runtime Environment using NetBeans IDE 5.0 platform. Each client has a main thread that manages the actions required by the user, as well as creating different threads for managing the connections that should be established (e.g. listening threads for providers). Clients are interconnected through a network that uses point-to-point TCP/IP communications implemented upon sockets.

A. Efficiency Analysis

Table II summarizes the time sequence, the number of cryptographic operations and the computational complexity for each stage of the protocol. Furthermore, we include, in the same table, the speed benchmarks corresponding to the cryptographic primitives used [7]. In summary, the computational complexity of evaluating the cryptographic operations required by our content authentication protocol is, at worst, $\mathcal{O}(|m|k^3 \log k)$, which linearly depends on the content size, and is bounded by a polynomial in the amount of k protocol participants. This cost represents the *worst* case, i.e. no signers are known and all the verifications must be performed, so that case must be seen as an upper bound. Indeed, as similar

Stage	No. crypto operations
1. C_m generation	$1H + 1S$
1.2. Signature process	$kS + kH + \frac{k(k+1)}{2}V + \frac{k(k+1)}{2}PI$
2. B's Checking on C_m	$1H + kPI + kV$
Subtotal:	$(k+1)(H + S + \frac{k}{2}V) + \frac{k(k+1)}{2}PI$
Worst represents:	$k(E + D + N) + 3(k+1)H +$
$\mathcal{O}(k \log k)$ [6]	$+(4k+3)S + (k^2 + 3k + 1)V$
Total Complexity:	$\mathcal{O}(\|m\|k^3 \log k)$

Legend: H: Hash generation; S: Signature generation; V: Signature verification; k: No. of signers; PI: Pathak and Iftode's protocol; E: Asymmetric encryption; D: Asymmetric decryption

Updated speed of cryptographic primitives benchmarks: `RSA 2048 Encryption`: 0.08ms/op; `RSA 2048 Decryption`: 2.78ms/op; `RSA 2048 Signature`: 2.78ms/op; `RSA 2048 Verification`: 0.08ms/op; `AES-256`: 113MB/s; `SHA-256`: 106MB/s

TABLE II
EFFICIENCY ANALYSIS (COMPUTATIONAL EFFORT.)

10 Signers			
	$m = 1$MB	$m = 10$MB	$m = 100$MB
Best	36.1460	321.8600	$3.1790{\cdot}10^3$
Worst	$1.3026{\cdot}10^3$	$1.5883{\cdot}10^3$	$4.4454{\cdot}10^3$
50 Signers			
	$m = 1$MB	$m = 10$MB	$m = 100$MB
Best	249.1860	$1.5739{\cdot}10^3$	$1.4821{\cdot}10^4$
Worst	$2.4964{\cdot}10^5$	$2.5097{\cdot}10^5$	$2.6421{\cdot}10^5$
100 Signers			
	$m = 1$MB	$m = 10$MB	$m = 100$MB
Best	695.4860	$3.3189{\cdot}10^3$	$2.9553{\cdot}10^4$
Worst	$2.3263{\cdot}10^6$	$2.3289{\cdot}10^6$	$2.3552{\cdot}10^6$

TABLE III
AVERAGE TIME OF EXECUTION (MS): BEST AND WORST CASE WITH 100% COLLABORATIVE NODES, I.E. ALL PARTICIPANTS COLLABORATE.

protocols implementing distributed authentication, the implemented public key authentication protocol is quite expensive in messaging cost. However, authors, in their analysis, try to reduce this cost using an epidemic algorithm called *Public key infection* for lazy propagation of protocol messages. By means of this improvement, the complexity gets reduced to $\mathcal{O}(k \log k)$. Readers interested in a more detailed description should refer to [6]. For instance, generating a certificate for contents between 1 and 100 MB takes less than 1 minute with the cooperation of 10 signers. As content's size and the number of signers increase, the computational cost increase significatively: A certificate for a content of 500 MB and 100 signers takes approximately 1 hour in the worst case, and less than 3 minutes when no signer is unknown. See, however, that this task is carried out just once, and that content access is considerably faster. On the other hand, at best, i.e. no instance of public key authentication is needed since signers are mutually known, generating a certificate for contents between 1 and 100 MB takes less than half a second with the cooperation of 10, 20 and 50 signers. The upper bound reached in simulations has been half a minute to sign a 100 MB content by 100 mutually known signers.

According to the communication overhead imposed by the proposal, signing stage incurs the major cost as well, i.e. $\mathcal{O}(k^3 \log k)$. The cubic factor results from the transmission of $\frac{k(k+1)}{2} k \log k$ messages for k participants in the public key authentication, at worst. Nevertheless, in the best case where all signers are mutually authenticated before, the number of messages decreases significatively to 19 messages, using 10 signers.

1) Average Time of Execution: Even though the use of digital signatures and encryption based on public key cryptography require extra computational cost in our proposal, there are two main responsible factors for the high computational complexity. First, the execution of several instances of public key authentication protocol increases the cost. Secondly, as the content certificate generation requires the cooperation of k participants, faulty, dishonest or malicious nodes, and network failures as well, involve additional complexity, as shown in Figure 2. In such a scenario, the total cost is comprised of an additional, wasted cost which is the result of the unsuccessful instances Δk. Thus, we can evaluate the decrease in the number of public key authentication instances with respect to the proportion of trustworthy peers which are prone to collaborate. Besides, this will allow us to analyze the impact that non-collaborative behavior (even churn problematic) has on the overall system. We then make a comparison between simulations of the content certificate generation where all participants know each other (best case) and simulations which require public key authentication (worst case). Moreover, we consider different sets of k participants (10, 50 and 100 signers), and several content lengths (1MB, 10MB and 100MB).

Table III summarizes the average time of execution in each case. We can derive some conclusions from these results. First, contents' size is actually an influent factor concerning time of execution due to its involvement in signatures by each and every k signers, which also carry weight in the total cost. Nevertheless, we can notice the growing differences between simulations with and without running the public key authentication protocol with respect to the number of involved nodes. For instance, generating a certificate for 1 MB contents, and with the participation of up to 100 signers takes less than one second in both cases. The cost increases for values of k up to 50 and having to authenticate public keys.

A key problem, however, is to guarantee that all the community members will collaborate to jointly carry out a process. We explore the probabilistic nature of non-collaborative behavior in a P2P community. Indeed, we simulate experiments based on some kind of conjecture over the collaboration nature of the whole community that one is immersed. The underlying idea is that a content owner will try to reduce the cost when she estimates it is highly possible to interact with non-collaborative peers of the community. These experiments work as follows. The owner first elaborates a conjecture $\zeta \in \{0.0, 0.1, \ldots, 1.0\}$ over the community's collaboration nature. All possible thresholds represent different community profiles, i.e. 0.2 means the existence of a 20% of cooperative nodes out of the total networking nodes in the community. At this point, the collaboration of a certain participant may be

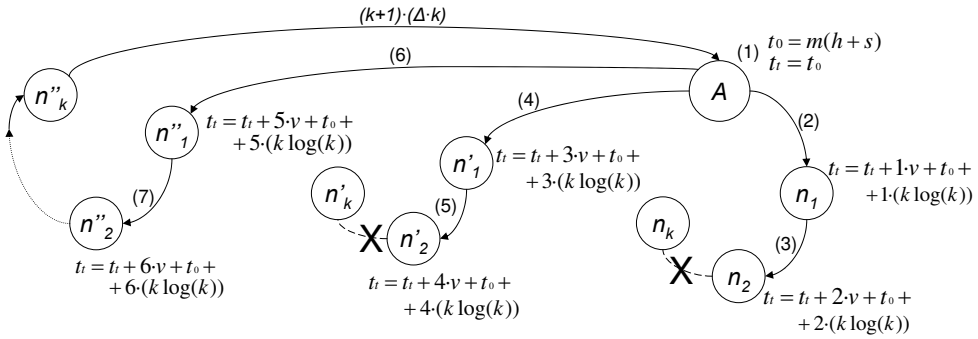

Fig. 2. Content certificate generation procedure when a signer does not collaborate: Time computation.

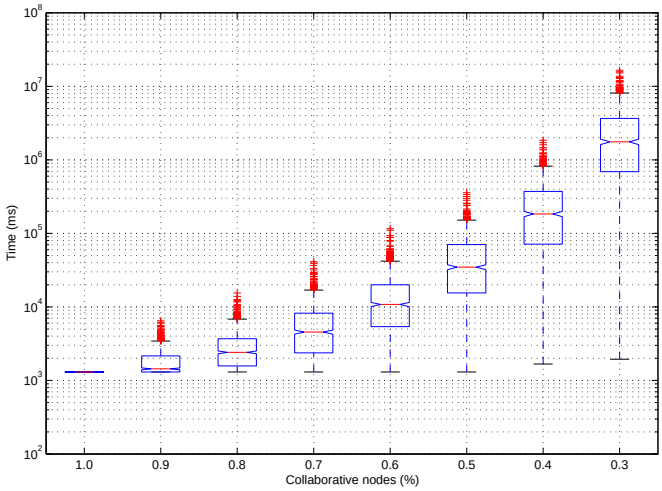

Fig. 3. Distribution of the time of execution for 1MB contents, 10 participants, and different percentages of collaborative nodes in the community at worst.

modeled using a uniformly defined value $\theta \in U(0,1)$. i.e. the probability at which a node is assumed to be collaborative. Thus, we experimentally estimate the likelihood of finding k collaborative (consecutive) nodes, i.e. k nodes whose value θ is above the given conjecture ζ set by the user, and jointly cooperate in signing the content certificate. So, if the computed value θ is higher than the threshold ζ, node n_i is supposed to be collaborative and, therefore, the protocol continues.

Experiments were conducted varying the maximum number k of required participants from 10, 50 to 100, and several content lengths. The experiment was run 1000 times for each threshold ζ. Some results were obtained: at best, with 90% of collaborative nodes out of 50, signing the content certificate takes 6 seconds on average, while with 10 participants and the same percentage of collaboration the average time of execution is 36 ms. Moreover, the fast-growing cost is clear when increasing the percentage of non-collaboration in the community: with 70% and 50 signers (known among them) we obtain a computational cost about $7.24{\cdot}10^4$ seconds.

We provide a characterization of the distribution of our results. The purpose of this is to show the most extreme values in the experimental data set (maximum and minimum values, which we can consider them as outermost situations or potential outliers), the lower and upper quartiles, and the median. The results are displayed as boxplots. For instance, Fig. 3 shows the computational cost (time of execution) distribution for the 1000 experiments run in each proportion of collaborative nodes in the community (from 100% to 30% of collaborative nodes) for 1MB contents, and 10 participants taking into account the complexity of the public key authentication. We can appreciate a "mild" variability of the simulated data, indicated by wider boxes. However, the 90% of collaboration gives us guarantees for, at worst, the cost of creating a 1MB-content certificate takes no more than 2 seconds with the cooperation of 10 signers which are unknown among them. On the other hand, though the unrealistic 40% of collaborative nodes also gives a less probable time of 2 seconds, the vast majority of the simulations ranges from 1.3 to 10 minutes, at worst. Similarly, experiments on the same scenario at best constitutes certainly an interesting result. Comparatively, the 90% of collaboration gives almost all computational cost values of less than 0.1 seconds, while the 40% less than 1 minute.

2) Determining k: This is a challenging problem, especially in a fully distributed, decentralized and dynamic system. A trivial way to solve it is by assuming or imposing a certain interval which limits the maximum and minimum number of signers according to the current community size. However, the highly transient community makes this implementation unrealistic in such environments. Applying the same ideas to the Byzantine type of failure [8], it is shown that $n > 3t + 1$ (where t nodes could be faulty and behave in arbitrary manners, and n is the total number of nodes) must hold even in the case of only needing to reach approximate agreement. As we showed before, the expected number of rounds to reach agreement may be exponential depending on the proportion of non-collaborative/faulty signers. However, some early works presented so far propose several theorems, based on probability, which guarantee the following [9]: (i) As long as a majority of the processes continues to operate, a decision will be made, (ii) if the number of faulty nodes is $O(\sqrt{N})$ then the expected number of rounds to reach agreement is constant, and (iii) if

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

TABLE IV
DETERMINING AMOUNTS OF k AT [BEST,WORST].

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

On the other hand, we may apply a similar approach, but backwards, as computing the average time of execution in order to estimate an appropriate number of signers according to the time a particular owner is willing to spend, i.e. how many signers the owner has to collet if she wants to spend no more than a certain period of time. We simulate, as in previous sections, different percentages of collaborative nodes in the community, and consider the best and worst case according to the requirement of instancing the public key authentication protocol. Table IV presents these results for 0.5 and 30 seconds. Note that, in the best case the number of nodes can be significantly higher, especially when having at one's disposal more than 30 seconds, e.g. we can use up to 400 nodes for generating a 10MB-content certificate in 30 seconds with mutually known signers immersed in a 60% collaborative community. On the other hand, we can consider 0.5 and 1 seconds as excellent lower bounds to generate 10MB(or less)-content certificates by collaborative communities and team-working. Moreover, 30 seconds may be a good mark to apply in common file sharing systems, even in the worst case.

3) Availability of Signers: The failure of a node/network link can be a serious problem as the proposed content authentication scheme depends on that network component for its execution. Many works have so far addressed availability for predicting per-node resource burdens and selecting appropriate data replication strategies. Some techniques basically lead to enhance routing schemes to route around faulty nodes or use multiple pass through the networks in order to avoid faulty networking elements, among others.

Here, we sketch some key ideas to deal with the discussion above. Particularly, faulty (maliciously or not) nodes incur additional concerns in content certificate generation and verification stages. In the former, unavailable signers imply new instances of the signing phase initiated by the owner, incrementing the whole overhead, as we state before. On the other hand, in the latter, at worst, a certain requester B, after a successful download, knows none of the signers' identities. If any signer n_i is (temporarily or persistently) unavailable, there is no way to authenticate its public key by means of the protocol adopted. A possible solution to this problem is based on the underlying use of gossip–based membership management, byzantine–tolerant information propagation, and/or authentication-free protocols. Although it is out of the scope of this paper, we refer the reader to the collection of papers in [10], [11], [12] for further details.

IV. CONCLUSION AND RESEARCH DIRECTIONS

In this paper we have analyzed a content authentication protocol based on Byzantine agreement, especially oriented to pure P2P systems. We have provided brief summary of the results in terms of the computational and communication overheads in tables and figures.

Future work includes several research lines. Both Sybil attack and the use of pseudonyms are challenging issues for certificate-based approaches. A Certification Authority can enforce randomly chosen node identifiers and limit each node to have only one alias. Nevertheless, the absence on such an entity generally forces the need of a consensus agreement which depends on the collaboration within a social group. This group, for example using threshold cryptography, can all together decide whether a new joining node is genuine or sybil. This is still an interesting open issue. A simple, but limited solution is by not allowing any member to revoke or change its identification item.

REFERENCES

[1] X. Zhang, S. Chen, and R. Sandhu, "Enhancing data authenticity and integrity in p2p systems," *IEEE Internet Computing*, pp. 42–49, November–December 2005.

[2] R. C. Merkle, "A certified digital signature," in *Proc. of the 9th Annual Int. Cryptology Conference on Advances in Cryptology*, 1989.

[3] R. Tamassia and N. Triandopoulos, "Efficient content authentication in peer-to-peer networks," in *Proc. of the 5th Int. Conf. on Applied Cryptography and Network Security*. LNCS, June 2007, pp. 354–372.

[4] Y. Desmedt, "Some recent research aspects of threshold cryptography," in *1st. Int. Workshop on Information Security, (ISW'97)*. Springer-Verlag, 1997, pp. 158–173.

[5] S. Farrell and R. Housley, "An internet attribute certificate profile for authorization," *RFC 3281*, April 2002.

[6] V. Pathak and L. Iftode, "Byzantine fault tolerant public key authentication in peer-to-peer systems," *Computer Networks*, vol. 50, no. 4, pp. 579–596, March 2006.

[7] "Speed benchmarks for some of the most commonly used cryptographic algorithms," Website, 2008, http://www.cryptopp.com/benchmarks.html.

[8] M. Barborak, A. Dahbura, and M. Malek, "The consensus problem in fault-tolerant computing," *ACM Comput. Surv.*, vol. 25, no. 2, pp. 171–220, 1993.

[9] M. Ben-Or, "Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols," in *Proc. of the 2nd annual ACM symposium on Principles of distributed computing*. ACM, 1983, pp. 27–30.

[10] A. J. Ganesh, J. Kermarrec, and L. Massoulié, "Peer-to-peer membership management for gossip-based protocols," *IEEE Trans. Comput.*, vol. 52, no. 2, 2003.

[11] K. Han, B. Ravindran, and E. D. Jensen, "Byzantine-tolerant, point-to-point information propagation in untrustworthy and unreliable networks," vol. 4658, pp. 207–216, 2007.

[12] W. Galuba, K. Aberer, Z. Despotovic, and W. Kellerer, "Authentication-free fault-tolerant peer-to-peer service provisioning," in *Proc. of the 5th Int. Workshop on Databases, Information Systems and Peer-to-Peer Computing*, Vienna, Austria, September 2007.