A Distributed and Adaptive Revocation Mechanism for P2P networks

Thibault Cholez, Isabelle Chrisment, Olivier Festor

A Distributed and Adaptive Revocation Mechanism for P2P networks

Thibault Cholez, Isabelle Chrisment and Olivier Festor
MADYNES - INRIA Nancy-Grand Est, France
{thibault.cholez, isabelle.chrisment, olivier.festor}@loria.fr

***Abstract*—With the increasing deployment of P2P networks, supervising the malicious behaviours of participants, which degrade the quality and performance of the overall delivered service, is a real challenge. In this paper, we propose a fully distributed and adaptive revocation mechanism based on the reputation of the peers. The originality of our approach is that the revocation is integrated in the core of the P2P protocol and does not need complex consensus and cryptographic mechanisms, hardly scalable. The reputation criteria evolve with the contribution of a peer to the network in order to highlight and help fight against selfish or malicious behaviours. The preliminary results show that the user perceived delays are not highly impacted and that our solution is resistant to reputation and revocation attacks.**

***Index Terms*—P2P networks, revocation mechanism, reputation mechanism, remote accounts, KAD**

I. Introduction

Peer-to-Peer (P2P) networks have proved their ability to gather and share a large amount of resources thanks to the collaboration of many individual peers. They are known to have many advantages compared to the client-server scheme: P2P networks scale better; the cost of the infrastructure is distributed and they are fault tolerant.

However, P2P networks encounter several difficulties induced by the growing number of malicious peers. The lack of central authority and the individual behaviour of the peers make it difficult for the P2P network to manage them. Malicious peers can be classified in three main categories; the malicious peer:

- does not follow the P2P protocol and tries to make attacks
- shares malicious content (malware, pollution, illegal content)
- behaves in a selfish fashion

Several studies have been made to measure the impact of these bad behaviours on the network. [1] and [8] monitored Gnutella and highlighted the tragedy of the commons, consequence of selfish behaviour: 70% of users do not share anything, 50% of resources are shared by only 1% of users. The authors warn against the limitation of spontaneous cooperation in anonymous groups and the possible collapse of such networks without real control mechanisms. The pollution phenomenon has also been studied by [12]; it appeared that, on average, 50% of the songs shared on Kazaa are polluted and even more of the newer files. Thus, malicious behaviours really degrade the quality of services proposed by public P2P networks.

In this context, P2P networks need a way to have the behaviour of their users supervised. We propose a fully distributed and adaptive revocation mechanism. Our architecture is designed for structured P2P networks which have proved their ability to be efficient [3] in their organisation and service offerings. The revocation is decided and adapted according to the reputation of each peer which is provided by remote accounts stored in a DHT (Distributed Hash Table). For the time being, the reputation evolves with the contribution of the peers to highlight selfish behaviours.

This document is structured as follows. Section II presents the related works on reputation and revocation within P2P networks. The foundations of our architecture are described in Section III which includes the concept of remote accounts used to store the reputation, the evolution of the reputation and the revocation mechanism. The revocation is further detailed and is illustrated in Section IV for the KAD network. Section V discusses a first performance evaluation and security issues. Finally, Section VI concludes the document and presents future works.

II. Related works

A. Reputation

Reputation management in a distributed environment is very challenging. The great majority of the reputation systems has indeed a local view, where each peer stores locally the reputation of another after having had some relationship with it [10] [6]. This approach has however several drawbacks. First, it is impossible to know if a peer is malicious before contacting it (feedbacks are not shared among the peers). Second, it is not adapted for large public P2P networks as the probability to meet a peer several times is so low that the reputation is inaccurate if it is used. That is why credit systems currently implemented in applications like eMule[1] can not fight against free riding; the local knowledge is not sufficient to determine if a peer free rides (few peers are known, few transactions are established with each one). The advantages are that the system scales well and it is suited for small communities of peers interacting frequently.

More recently, the concept of remote accounts presented in PeerMint [7] can lead to another solution for reputation management. The idea is the following: each peer has a public account (i.e. an information set) stored in the P2P network. The storage of an account is done by mapping it

[1] Description of eMule credit system: http://www.emule-project.net/home/perl/help.cgi?l=1&rm=show_topic&topic_id=134

on a set of peers thanks to the DHT used by structured P2P networks (Chord, Pastry, Kademlia...). This set of peers is periodically renewed to keep the information in the network despite churn; moreover, replication makes the mechanism more reliable. The remote accounts allow to build a global reputation management system where the reputation of each peer is stored in the DHT and accessible to the others.

B. Revocation

Designing a revocation mechanism adapted for P2P networks is difficult. The first way to revoke a peer is to build an access control system. The control cannot be made by a central authority because it is not adapted in a P2P environment, so it is the responsibility of the network to enforce control in a distributed way. In [11] and [14], the authors present and experiment different approaches to achieve admission control in a peer group. Several policies are possible: the new peer must gather the agreement of a fixed number of peers, or a number proportional to the size of the group. The second proposal [14] evaluates the performances of cryptographic mechanisms used to implement the admission control. It appears that they scale badly and are more suited for ad-hoc networks with high security requirements than for large public P2P networks.

In [5] an original way to achieve dynamic revocation in a P2P network is presented. When a peer detects that another is malicious, it sends a revocation notification that includes the malicious peer and itself, considering that its own life is less important than the goodness of the network. Therefore, it prevents the revocation mechanism to be hijacked because the cost to revoke is very high. However, this mechanism has important limitations. In fact, it can only be used whithin a private network but not in a public one where each peer has individual interests.

III. General Architecture

A. Remote Accounts

In our system, we use remote accounts because they are efficient to introduce reputation in P2P networks. It is a way to adapt a centralised reputation system (for example eBay) to a decentralised network. With this system, each user has a grade evolving with the feedbacks of the others, so that each knowledge is shared with the community.

Stored in the DHT, each peer's account has a logical address which must remain unchanged after each session in addition to the peer's address itself. The application eMule already uses two identities for the peers of the network. The first called clientID (or KadID) is the 128 bits address of a peer in the DHT and is randomly chosen at the first connection. The second is called userID and results from a hash of the computer. This address is used for the credit system and public/private keys are associated to it to ensure the identity of the peer claiming a userID. Our solution links the peer's account to the userID as presented in figure 1. In this way, the userID is not used to localy store the credits of a peer but provides an entry in the DHT where to store its public account.

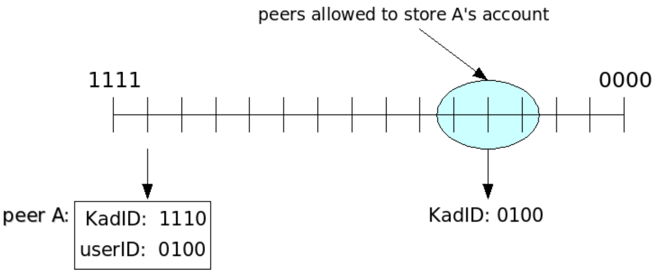

Fig. 1. Account storage in the DHT

Conflicting reputation references are avoided by making a lookup for its own ID before creating the associated account.

An account just contains few data, that is easily storable even with replication:

- userID (128 bits) : place of the account in the DHT
- publicKey (128 bits) : the account's owner has the associated private key
- trustRating (16 bits) : reputation of the account's owner
- blackboard (few kiloBytes) : displays the current transactions of the account's owner

B. Evolution of Reputation

In a file sharing application, the evolution of the reputation concerns the way a peer contributes to the network, increasing when it uploads data and inversely, decreasing when it consumes resources. Thus, users are motivated to share data which are interesting for the community. In parallel, existing mechanisms ensure that rare data are sent with priority. With such a reputation rate, identifying free riders becomes easier. The major difficulty consists in finding a secure way to create and update the reputation. When a peer joins the network for the first time, it receives an initial positive reputation allowing it to start the first transactions. This initial reputation is needed to initiate transactions inside the network. No initial positive reputation would result in a global deadlock, like an automaton without a token.

Next, the evolution must be based on the fact that a transaction always involves two peers which exchange the same amount of data in opposite directions. During a transaction, both peers have to write the exchange on a part of their account, we called *"blackboard"*. A blackboard's entry displays information for each transaction in progress: the partner in the transaction, the exchange direction, the amount of data sent or received (periodically updated). At the end of the transaction, the peers in charge of the accounts have to update the reputation according to the information displayed on the blackboards. To prevent a collusion of malicious peers which could display false transactions in order to increase their reputation, peers in charge of the accounts can not trust directly the information sent by the involved peers. They have to communicate among themselves to check if the same announcements have been received by the other part to check the consistency. This condition ensures that the transaction

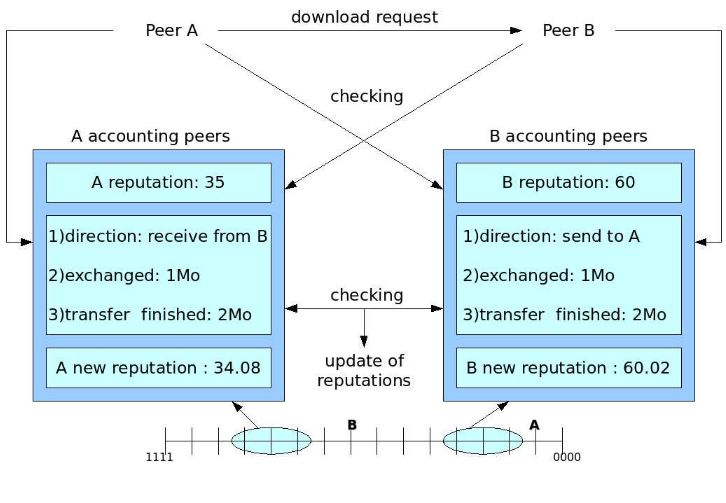

Fig. 2. Accounts usage during a transaction

is reflected by both peers with the same ratio and avoids the hijacking of the mechanism. Each bonus of reputation resulting from a transfer has its opposite. Figure 2 presents the usage of remote accounts during a transaction and the evolution of the reputation as a consequence.

In P2P networks, the question of privacy is crucial. The reputation mechanism does not set up a new menace for the private life of users. The reputation grade is just a ratio between downloaded and uploaded data. Considering the grade, it is not possible to infer the activity of a peer, but only if the activity is balanced or not. Moreover, it is not possible to deduce from the blackboard which file is being transferred because several transactions are needed to retrieve a complete file.

C. Revocation Mechanism

The revocation mechanism uses the reputation displayed on the account of each peer to decide if, and how, a peer must be revoked. The reputation can evolve until a threshold triggering the revocation. As P2P networks are based on individual peers serving each other, a way to revoke a peer in a fully distributed manner is to check whether the requesting peer is worthy of receiving the service before providing it. If all the peers of the network check the reputation before providing a service, a peer with a bad reputation is automatically revoked, its requests being refused by the network. Moreover, this mechanism is adaptive because the refused services can change according to the different criteria of reputation used (contribution, quality of shared content...). The services provided by a P2P networks are generic: a bootstrapping process, a publication process (indexation of the shared files in the network), a search engine, and direct connections to download and upload data.

The idea of adapted sanctions has been presented by [9]. The authors describe three levels of counter-action according to the level of free riding detected: decrementing TTL, ignoring requests and disconnecting the malicious peer. In our solution, each service can be checked independently. When a peer only has a bad sharing ratio, it is relevant to remove its rights to download data but it is not necessary to remove the other services needed to participate to the network. So, this peer will be able to download again after having shared more resources. On the opposite, when a peer is revoked for security reasons, all its rights must be removed in order to exclude it entirely from the network (see table I).

Revoked Services	Sharing	Security
bootstrap and routing table	No	Yes
publication and upload	No	Yes
download	Yes	Yes
search	No	No

TABLE I
RELEVANCY OF THE REVOKED SERVICES ACCORDING TO THE REPUTATION CRITERIA

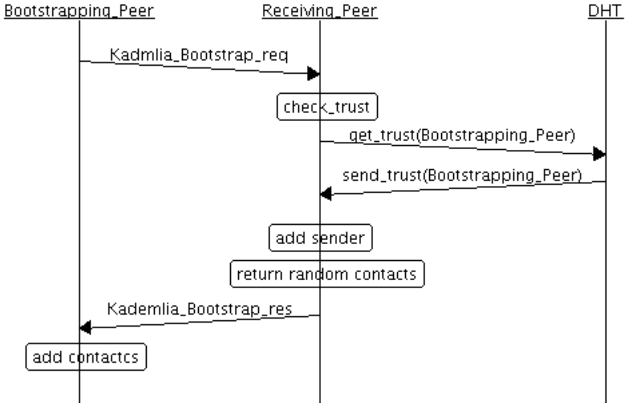

Fig. 3. Reputation check during the bootstrapping process

IV. DESIGN FOR THE KAD NETWORK

This section explains how and what services can be revoked, taking example of the KAD network. KAD is a part of the popular eMule and aMule file-sharing applications. It is based on the Kademlia protocol [13] and is one of the widest deployed structured P2P network with millions of simultaneous users.

A. Bootstrapping Process

This phase is necessary to join the network. Concretely, the bootstrapping peer asks another peer to send it other contacts from the network to initialise its routing table and inversely, to be referenced by other peers. As a first step of the revocation mechanism, the receiving peer will have to check the reputation of the bootstrapping peer before sending its contacts. This process is illustrated in figure 3. Unfortunately, a malicious peer can become a bridge for revoked members by avoiding the check step. Therefore, controlling the bootstrapping process allows to quickly carry out some total revocations. Controlling the other services allows overcoming the previously described weakness and refining the sanctions.

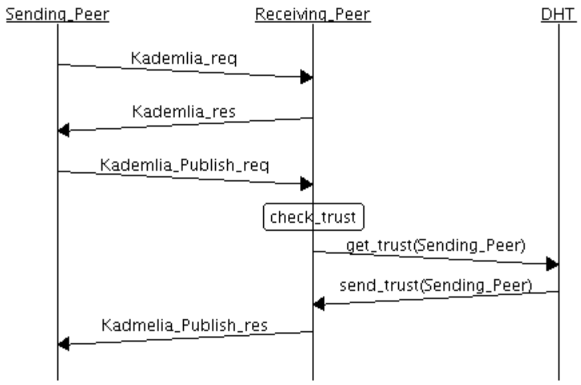

Fig. 4. Reputation checking during the publication process

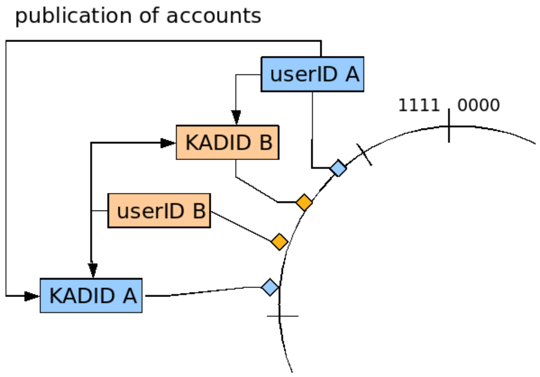

Fig. 5. Publication of accounts between modified KAD clients inside a tolerance zone

B. Other Services

When a peer is connected to the network, services (publication of contents, search, data download...) are achieved by sending requests to the other peers. In Kademlia [13] this is done in two phases. Firstly, *Kademlia_REQ* are sent to find nodes which are potentially able to deliver the service (according to their place in the DHT). This phase is general and only concerns the iteration mechanism used to find several peers in a part of the P2P network. In the second step, when the nodes are found, a specific request is sent to ask for a particular service. The reputation checking must be done before the specific request for three reasons. Firstly, a peer can be a bridge and search contacts for other uncontrolled peers. Secondly, the real services are provided by the specific requests. Controlling the reputation at this point allows to revoke independently the different services. Finally, checking the reputation for *Kademlia_REQ* would increase the overhead for no advantage. The figure 4 presents the running of a publication request and includes the reputation checking. The other requests (search, data download) follow the same scheme.

However, inserting the revocation mechanism into the search function is not relevant for several reasons. Firstly, it is not a service through which a peer can damage the network. Then, searching is useless when the other services are inactive behind. Finally, it would also introduce overhead to the network and unnecessary delay for all users.

C. Implementation

We have implemented the revocation mechanism in KAD. To do that, we have introduced different modifications in the KAD client:

- our modified client can manage a new kind of information called ”Account”;
- the associated requests search/store Account were written, which partially behave like existing requests on keywords, files or notes;
- new functions were added in the UDPListener, where all network’s requests are processed. In fact, these functions do the service-oriented revocation, searching for the reputation and checking it.
- KadID and userID allocation were cheated to control the place of the peers in the DHT.

We have tested that the reputation storage and retrieval and the revocation of services work fine on the modified peers. However, as our implementation defines new data types and messages to manage the accounts, we can just test our revocation mechanism on a few peers. Presently, as we test the mechanism with few resources, we have defined the KadID and UserID of modified peers to place them in the same tolerance zone. In this way, we are sure that the accounts can be stored (figure 5) when using the KAD publication mechanism. That is enough to verify the mechanism, but without the real number of replicated peers, performance measures would be wrong. That is why we are going to scale up our testbed on EmanicsLab to measure performances and compare the results with the evaluation presented in section V.

V. Analysis and Discussions

A. Performances Evaluation

The thesis [2] has led a performance evaluation of the KAD network which allows us to discuss some a priori performance results. The average delay needed to store information in the network is about 200 seconds. This time is needed to find ten peers (for the replication) with a KadID close to the hash of the information to store. This delay will occur the first time that a peer connects itself to the network and periodically later to maintain the account in the network despite the churn. The delay to retrieve the information fluctuates and depends on the replication. The more replication there is, the more robust the stored information is and the quicker it is retrieved. The information is retrieved linearly, so the more a peer waits, the more results are returned. A 100 seconds delay seems to be sufficient to retrieve enough information to guess the real reputation of a peer. This delay could seem huge because it appears prior to each service of the network, except the search as explained. But in fact, 100 seconds to bootstrap are not penalising, the publication process is entirely transparent for the user and 100 seconds preceding a download are insignificant regarding the average waiting time spent in download queues. These elements show that the resulting delays would not be sensed by the users; this will be confirmed by the implementation.

B. Security Issues: Case Study

Security issues are a major constraint when designing such mechanisms. We have tried to anticipate the possible malicious behaviours of each actor to make the mechanism resistant to attacks.

1) Accounting and Reputation Attacks: The first interest of a malicious peer is to avoid its reputation to decrease when it downloads, and to increase its reputation more than allowed when it uploads. To do this, some malicious peers will try to modify the information displayed on the blackboard at the end of the transaction. In the first case, the protocol does not allow to decrease the amount displayed on the blackboard because this action has no meaning during a transfer. In the second case, there would be a disagreement between the two blackboards and the reputation must still be updated otherwise the mechanism would be easily hijackable. In case of disagreement, the value which must be used to update both reputations is the one displayed by the downloading peer because this value can not be decreased and the downloading peer has no interest to increase it (its reputation would decrease more than needed). An agreement can occur between two peers but only one will gain reputation against the other.

It is also possible that a peer in charge of the account of another lies when the reputation is requested. This behaviour does not have a lot of consequences because of the replication. A reputation's request will always get several responses. As the majority of the peers are supposed to be honest, it is simple to retrieve the right value among the responses using majority decisions. It also prevents the mechanism from byzantine failures.

However, the initial reputation could become a problem if the hash function giving the userID (ie a new account) is hijackable. In fact, a malicious user could create and use a new account when its initial reputation is over or transfer the reputation of the new account to the main via fake transactions; but a possible solution is described further.

2) Revocation Attacks: The revocation mechanism is robust because it is fully distributed. If a malicious peer decides to bypass the protocol, answering to revoked peers or ignoring good peers's requests, will have a very limited impact. The revocation is assured because all the peers of the network refuse to serve revoked peers; one individual action (whatever it is) has no consequence for the mechanism.

But there is still a way to hijack the revocation mechanism. It consists in a coalition of peers placed in the same point of the DHT, so that they are able to take in charge the majority of the replicated account of a peer. If an account is replicated *n* times, placing such (*n/2 +1*) peers in this way can make the entire network revoke the victim peer if the malicious peers lie together. If we consider the following equations:

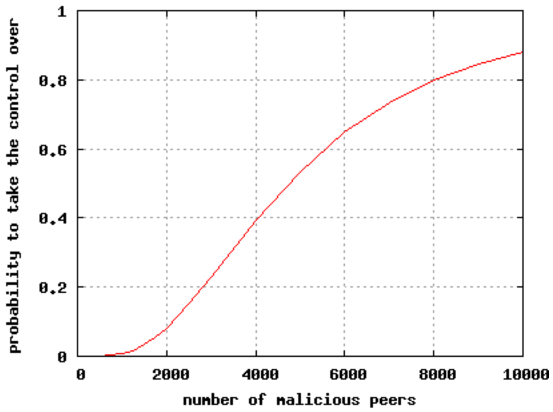

Fig. 6. Probability to take the control over an account in function of number of malicious peers

$$P(X=i) = \frac{C_x^i * C_{4000}^{10-i}}{C_{4000+x}^{10}} \qquad (1)$$

$$P(X \geqslant 6) = \sum_{i=6}^{i<=10} P(X=i) \qquad (2)$$

The probability (2) is based on the indexation mechanism of KAD which uses a tolerance zone of 8 bits defining which peers are able to store an information (on average 4000 peers for 1 million of unfirewalled users). Let x be the number of malicious peers in a zone; the probability to take the control over an account is equivalent to take control on a defined number of malicious peers until having more than the half of the total number of requested peers for the replication (by default 10). The probability to choose i malicious peers among 10 is represented by the hypergeometric law with parameters (10, x, 4000+x) (1). The attack is successful when at least 6 peers are taken.

The graph 6 shows the probability that the attack be successful regarding the number of malicious peers. We can see that the tolerance zone in which an account is stored (regarding the ID) is big enough to make a total control difficult without at least thousands of peers because the storage process is not entirely deterministic (with a number of controlled peers greater than 10000 the probability is close to one). Therefore, if we make the assumption that the possibility to obtain many KadID is limited (basically one ID per IP address or computer), this attack becomes very unlikely given the resources needed.

But the current implementation in KAD is clearly insufficient and permits to announce up to 2^{16} fake peers in the same zone [17]. However, several propositions have been investigated to limit the Sybil attack. In [17], the authors propose a central authority (CA) delivering KadIDs by cell phone. The KadID is encrypted by the CA private key from an IP address and an expiration time and can be decrypted by any peer with the CA public key to check the validity of a KadID. Castro et al. [4] presented another design of trusted certification authorities to secure peer joining. We can also consider with interest the project Keypeer [18] which develops a certificate authority delivering keys over a DHT. Finally, KadID and userID allocation can be linked. When KadIDs are distributed by an authority (either centralized or distributed) but not chosen by the client, the hijacking of userIDs (to retrieve reputation) is also resolved. In fact, with a stronger KadID, the userID can be derivated from it by a public function instead of being calculated separately.

It is also possible to detect the peers participating in an Eclipse attack and to revoke them to stop it. The detection has been investigated by [15] and [16]. They observe via anonymous auditing the bounding degree of a node to check if it is involved in the attack or not.

VI. Conclusion

Supervising the behaviours of the malicious users is the key for the good development of public P2P networks. Regarding the weakness of the currently implemented incentive mechanisms and the consequences of malicious and selfish behaviours on the networks, it is important to make the behaviours of the peers suit with the P2P philosophy. To answer to this major problem, we have proposed a revocation mechanism which is fully distributed, adaptive, and which does not require distributed consensus or complex cryptographic mechanisms. The revocation is not a layer above the network but is inserted in the core of the protocol, supervising the services provided by the P2P network. The reputation needed to take decisions is provided by remote accounts based on the DHT used by structured P2P networks. For the time being, the evolution of the reputation concerns the way the peers contribute to the network (regarding the bandwidth usage) to fight against the tragedy of the common problem. Each transaction between two peers is temporarily displayed on their blackboard to prove the evolution of the reputation. First analysis show that the resulting delays should not be sensed by the users. Moreover, the case study done to inventory the attacks tends to prove that the system itself is robust, as long as the allocation of a peer's ID is secured.

The future works consist in scaling up our testbed to evaluate the performances of the mechanisms implemented on KAD, particularly concerning the delays and the overhead. In a next step, we will develop the reputation mechanism to take into consideration new criteria such as the quality of the shared content to fight against pollution and malware diffusion, or to detect and revoke peers participating in a Sybil attack.

References

[1] Eytan Adar and Bernardo A. Huberman. Free riding on gnutella. *Journal First Monday*, September 2000.

[2] Rene Brunner. A performance evaluation of the Kad-protocol. Master's thesis, University of Mannheim and Institut Eurecom, 2006.

[3] Miguel Castro, Manuel Costa, and Antony Rowstron. Debunking some myths about structured and unstructured overlays. In *NSDI '05: Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation*, Boston, MA, USA, May 2005.

[4] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach. Secure routing for structured peer-to-peer overlay networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):299–314, 2002.

[5] Jolyon Clulow and Tyler Moore. Suicide for the common good: a new strategy for credential revocation in self-organizing systems. *SIGOPS Oper. Syst. Rev.*, 40(3):18–21, 2006.

[6] Elizabeth Chang Farookh Khadeer Hussain and Omar Khadeer Hussain. State of the art review of the existing bayesian-network based approaches to trust and reputation computation. In *ICIMP 2007: The Second International Conference on Internet Monitoring and Protection*, July 2007.

[7] David Hausheer and Burkhard Stiller. Peermint: Decentralized and secure accounting for peer-to-peer applications. In Raouf Boutaba, Kevin C. Almeroth, Ramn Puigjaner, Sherman X. Shen, and James P. Black, editors, *NETWORKING*, volume 3462 of *Lecture Notes in Computer Science*, pages 40–52. Springer, 2005.

[8] Daniel Hughes, Geoff Coulson, and James Walkerdine. Free riding on gnutella revisited: The bell tolls? *IEEE Distributed Systems Online*, 6(6):1, 2005.

[9] Murat Karakaya, Ibrahim Korpeoglu, and Ozgur Ulusoy. A distributed and measurement-based framework against free riding in peer-to-peer networks. In *P2P '04: Proceedings of the Fourth International Conference on Peer-to-Peer Computing*, pages 276–277, Washington, DC, USA, 2004. IEEE Computer Society.

[10] Mujtaba Khambatti, Partha Dasgupta, and Kyung Dong Ryu. A role-based trust model for peer-to-peer communities and dynamic coalitions. In *IWIA '04: Proceedings of the Second IEEE International Information Assurance Workshop*, page 141, Washington, DC, USA, 2004. IEEE Computer Society.

[11] Yongdae Kim, Daniele Mazzocchi, and Gene Tsudik. Admission control in peer groups. In *NCA '03: Proceedings of the Second IEEE International Symposium on Network Computing and Applications*, page 131, Washington, DC, USA, 2003. IEEE Computer Society.

[12] J. Liang, R. Kumar, Y. Xi, and K. Ross. Pollution in peer-to-peer file sharing systems. In *IEEE Infocom*, pages 1174–1185, march 2005.

[13] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 53–65, London, UK, 2002. Springer-Verlag.

[14] Nitesh Saxena, Gene Tsudik, and Jeong H. Yi. Admission control in peer-to-peer: design and performance evaluation. In *SASN '03: Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, pages 104–113, New York, NY, USA, 2003. ACM Press.

[15] Atul Singh, Miguel Castro, Peter Druschel, and Antony Rowstron. Defending against eclipse attacks on overlay networks. In *EW11: Proceedings of the 11th workshop on ACM SIGOPS European workshop*, page 21, New York, NY, USA, 2004. ACM.

[16] Atul Singh, Tsuen-Wan Ngan, Peter Druschel, and Dan S. Wallach. Eclipse attacks on overlay networks: Threats and defenses. In *INFOCOM*, 2006.

[17] Moritz Steiner, Taoufik En Najjary, and Ernst W Biersack. Exploiting KAD: possible uses and misuses. *Computer communications review, Volume 37 N5, October 2007*, 2007.

[18] Rita H. Wouhaybi and Andrew T. Campbell. Keypeer: A scalable, resilient distributed public-key using chord. columbia university. Technical report, Columbia University, November 2005.