

SuperTrust – A Secure and Efficient Framework for Handling Trust in Super Peer Networks

Tassos Dimitriou, Ghassan Karame and Ioannis Christou
Athens Information Technology
19.5km Markopoulo Ave., 19002, Peania
Athens, Greece {tdim,gkar,ichr}@ait.edu.gr

Abstract

In this paper, we describe SuperTrust (Super Peer Trust Handling), a secure framework designed to handle trust relationships in Super peer networks. What distinguishes SuperTrust from other contributions is that trust reports remain *encrypted* and are never opened during the submission or aggregation processes, thus guaranteeing privacy and anonymity of transactions. As reputations of peers influence their future interactions, we argue that such systems must have properties like fairness and soundness, persistence, eligibility and unreusability of reports, perhaps similar to the properties of current electronic voting systems.

SuperTrust is a *decentralized* protocol, based on K -redundant Super peer networks, that guarantees the aforementioned properties and is in some sense complementary to the models proposed for building trust among peers. Additionally the framework is very efficient and minimizes the effects of collusion of malicious Super peers/aggregators. We have tested the framework on a large subset of peers and demonstrated via simulations its superior performance with respect to network stress and response time, when compared to the other proposed protocols.

1 Introduction

Peer-to-peer (P2P) systems have recently become a popular medium that facilitates sharing huge amounts of data (e.g: KaZaA[1], Gnutella[2]). P2P networks distribute the main costs of sharing data, bandwidth, disk space for storing files and offer improved autonomy and reliability. These incentives behind peer-to-peer architectures derive from their ability to function and scale in the presence of high dynamism and in spite of failures without the need of a central authority.

P2P systems can be categorized based on their degree of centralization. In pure systems such as Gnutella, all peers share equal roles and responsibilities without the presence of a central authority as opposed to hybrid systems where a central server manages locating resources. However, both pure and hybrid architectures suffer from several problems ranging from reliability issues to threats on availability. The wide success of KaZaA has driven the attention towards another promising architecture: *Super Peer systems*. The latter combines the advantages of pure and hybrid systems as a mean of addressing the problems induced in both architectures by offering increased scalability and promises for enhanced security.

Other P2P architectures provide for anonymity when searching for content (Gnutella, Freenet[3]). However, sharing files with anonymous and unknown users challenges the very notion of systems security. Genuine looking files may contain viruses or self-replicating software which can destroy data and cause massive damage. Spyware or Trojan programs can be downloaded helping release sensitive information or allowing an attacker to access a computer at will. This lack of accountability opens the door to malicious users abusing the network and hinders the promotion of P2P systems in more useful settings.

Due to the absence of methods to convince a peer that another node is malicious, the need for trust in P2P systems emerges as one of few available mechanisms to keep malicious nodes from damaging the network. According to a study performed in [4], even the simplest reputation systems can provide a factor of 4 to 20 improvement in performance over no reputation system. In this respect, peer reputations present themselves as essential security tools to protect good peers from malicious ones.

These facts lead to a widespread use of reputation management systems, e.g. eBay[5]. In fact, many researchers ([6, 7, 8, 9, 10, 11]) have proposed the use of reputation mechanisms as a tool to evaluate the trustworthiness of peers. Each peer is associated with a “reputation value” that indicates how trustworthy the peer is. Interacting peers rate their performance and form an opinion of each other. Hence these metrics really depend on *peer reviews* by which peers slowly develop their digital reputations.

In this work, we are *not concerned with developing a new reputation system or model for building trust among peers*. Instead, we focus on developing an efficient and secure framework for distributing and accessing the trust ratings in a way that preserves the peers’ *privacy* and *anonymity* of transactions by taking advantage of Super Peer architectures. This suggests that peers who rate other peers must remain anonymous and the actual ratings must remain secure. Furthermore, we argue that nothing must affect the submission process: Peers must not be able to affect the system if they submit wrong values, colluding peers must not be able to alter the resulting ratings and the contents of the ballots must not be visible even by trust handling peers.

Our contribution in this work is threefold: First, we introduce SuperTrust (Super peer Trust Handling), a novel framework that makes use of efficient cryptographic tools as a mean of adding security, fairness and soundness to trust handling issues in Super Peer networks. SuperTrust takes advantage of the peers’ heterogeneity in such networks in order to manage and distribute trust values across peers. In addition, we make sure that properties like *anonymity*, *privacy* and *integrity of reports*, *fairness* and *soundness*, etc., are preserved. Second, we analyze SuperTrust against a multitude of attacks that seem important in designing new protocols for submitting trust reports and we identify anonymity and protection against collusion as one of the key challenges in the area. Finally, we present an evaluation of our work derived from a Java implementation of SuperTrust. This assessment relates to the system’s responsiveness, performance, resilience to malicious behaviors and network stress.

2 Design Goals

In this work we treat the reports that peers submit in order to rate their interaction with other peers as objects that deserve the same attention ballots receive in voting systems [12]. We present the first precise definition of design properties that *any* protocol for handling trust values must satisfy. We also present a protocol that can be shown to fulfill this list of desired properties. It is hoped that this new protocol can demonstrate the usefulness and viability of the provided framework, thus facilitating further research in the area.

- *Anonymity*: The system should support the peers' right to secrecy of their reporting ratings. Hence peers should not lose their identities because their expressed opinions have been revealed, accidentally or not. When this happens, a malicious node may try to eliminate these peers by mounting denial of service attacks against them.

There is also another side to the anonymity coin. Since trust reports will reside in the system and stored in peers themselves, it is also imperative to protect the reporting peers from being victimized by targeted attacks by malicious nodes who want to prevent them from sending the actual trust ratings to requestors of this information.

- *Privacy and Integrity of reports*: An opinion expressed by some peer in the form of a report should be protected from disclosure and modification attacks. A malicious node or a *collusion* of such nodes should not be able to eavesdrop or modify these ratings at will.
- *Persistence*: All trust reports should be accounted in building the reputation of a peer even when these reporting peers are no longer in the network. Any other option could be potentially exploited by a set of malicious peers who are always in the system, sending the same bad votes or masking legitimate peers from expressing authentic opinions.
- *Fairness and Soundness*: No one should be able to change, add, or delete trust reports without being discovered. The final computed rating must be perfect, either because no inaccuracies can be introduced or because all introduced inaccuracies are detected and corrected. This should be the case even when colluding authorities exist in the system.
- *Eligibility*: Only peers legitimate to express an opinion about some other peer should do so. This protects from malicious nodes giving poor ratings for peers that have never interacted with. For this to happen we require that when a peer u interacts with a peer v , there should be some *proof* of this interaction. It is only then that u can submit a report for the service offered by v .
- *Unreusability*: In a complementary sense we should prevent report stuffing. A peer eligible to express an opinion should be able to do this only *once* for a particular interaction it had with v . Further submissions should be detected and not be used to update the trust value of another peer.
- *Verifiability*: Any peer should be able to verify that all trust reports have been accounted correctly for the final rating of some peer v . We require verifiability to be a system property so that trust values reflect the actual reputation of any peer and that reports are never left out of consideration, accidentally or not. Furthermore, we require that this property is true even though trust reports remain encrypted.
- *Efficiency*: The entire process should be as efficient as possible. This includes computation of the final trust value of a peer v based on reports by other peers, or the actual process of submitting a report by peer u .

3 Related Work

While there are many papers that deal with models and architectures for building reputation and trust among peers (see for example [6, 7, 8, 9, 10, 11]), there have been only a few works that attempt to secure the process of submitting or retrieving trust reports. An even smaller subset of these contributions were proposed for Super peer networks. Furthermore, it is worth emphasizing that while these works use voting terminology (ballots, votes, voters, etc.) they fail to provide all the requirements mentioned in the previous section. In what follows, we will briefly describe those contributions that are related to our work, in both pure P2P and Super peer models.

Kamvar *et al.*[8] attempt to secure the computed reputation values by having the trust value of a peer v be computed by a set of *mother peers* which are selected by a *deterministic* process based on the identity of v . A peer u , requesting the trust value of peer v , first queries all of v 's mother peers and then takes a majority vote. Unfortunately, this system lacks anonymity, it can fall prey to peers presenting multiple identities [13], and its transactions between a querying peer and the mother peers are not secured.

TrustMe, presented by Singh and Liu [14], follows a similar approach in the sense that trust values for v are stored in *trust-holding agent (THA)* peers that are randomly distributed in the network and not known to v . Unfortunately, when peer u submits a trust rating to one of the THA peers of v , its rating of the interaction with v is revealed, which is a serious weakness of the protocol since trust reports are protected from disclosure and modification attacks only by *outside* attackers and not by malicious THAs. In a perhaps complementary way, u can never be sure that its trust report was taken into consideration. Thus this protocol also lacks the verifiability and fairness properties mentioned in Section 2.

PeerTrust[15] is a framework that includes an adaptive trust model for quantifying and comparing the trustworthiness of peers based on a transaction-based feedback system over a structured P2P overlay network. This framework focuses on providing confidentiality and integrity but lacks many of the other required properties mentioned in the previous section.

Despite the various proposals for reputation based systems in the literature, very little work is done in the context of Super peer networks which are considered nowadays the most promising taxonomy with respect to security and scalability. Cornelli *et al.*[16] propose to base reputation sharing on distributed polling whereby a requestor u can assess the reliability of perspective providers by polling its peers. These peers respond to the poll with the reputation of each offerer. The votes are then forwarded to u , giving it the opportunity to choose the most reputable peer. In [17], this work has been extended to cover Super peer networks. The only difference is that the designated Super peers accumulate the votes and forward them to u to make its decision.

Unfortunately, in both works u must individually contact *each* voter and ask for confirmation in order to rule out faked votes (loss in efficiency). It is also exactly at this point where peers lose their anonymity since a malicious peer can query for its own trust value in order to identify the voters who give poor trust ratings. Another drawback of the system is that votes do not have persistence as only online voters can give back reports.

Mekouar *et al.*[18] proposed a reputation management scheme for partially decentralized peer-to-peer systems. The scheme aims to build trust among peers based on their past experiences and feedback from other peers. Supernodes update and store the reputation results of the peers lying within their designated cluster. In addition, two selection advisor algorithms are used to help select the most trustworthy peer from which to download. However, this work lacks anonymity and security since it does not provide any measure that secures the various voting stages.

A comparison of these protocols and their characteristics is shown in Table 1. To be fair, however, we need to mention that these protocols were not designed with the properties of Section 2 in mind. A “N/A” indication shows that a property cannot be directly deduced/enforced or that is not part of the security protocol but perhaps of a broader one. A “Partial” characterization indicates that the corresponding property is not entirely achieved.

Table 1: Comparison of protocols

Property/Protocol	Kamvar <i>et al.</i> [8]	Cornelli <i>et al.</i> [16]	Chhabra <i>et al.</i> [17]	Singh and Liu [14]
Anonymity	No	No	No	Partial
Privacy	No	Partial	Partial	Partial
Persistence	No	No	No	Yes
Fairness	No	No	No	No
Eligibility	N/A	N/A	N/A	Yes
Unreusability	N/A	N/A	N/A	Yes
Verifiability	N/A	N/A	N/A	No
Efficiency	N/A	No	No	Yes

4 The SuperTrust Framework

We start by presenting an outline of our framework: SuperTrust is designed for K -redundant Super peer networks. In traditional Super peer networks, each peer can be either a super node (SN) or an ordinary node (ON). TCP connections are therefore established between ON and its SN and between pairs of SNs. The SN tracks the content of its designated ONs. On the other hand, K -redundant Super peer networks take benefit of having K -super peers working in a round robin fashion for each cluster (a cluster is the grouping of the leaf nodes of each super peer) in order to ameliorate the performance of Super peer networks. In fact, super peer redundancy decreases the load on super peers and tolerates more failures[19]. Our framework will make use of such benefits for the purpose of enhancing security in the voting process. Figure 1(a) depicts a 2-redundant super peer network. Dark nodes represent Super peers, white nodes represent ordinary peers and the three clusters are marked by dashed lines.

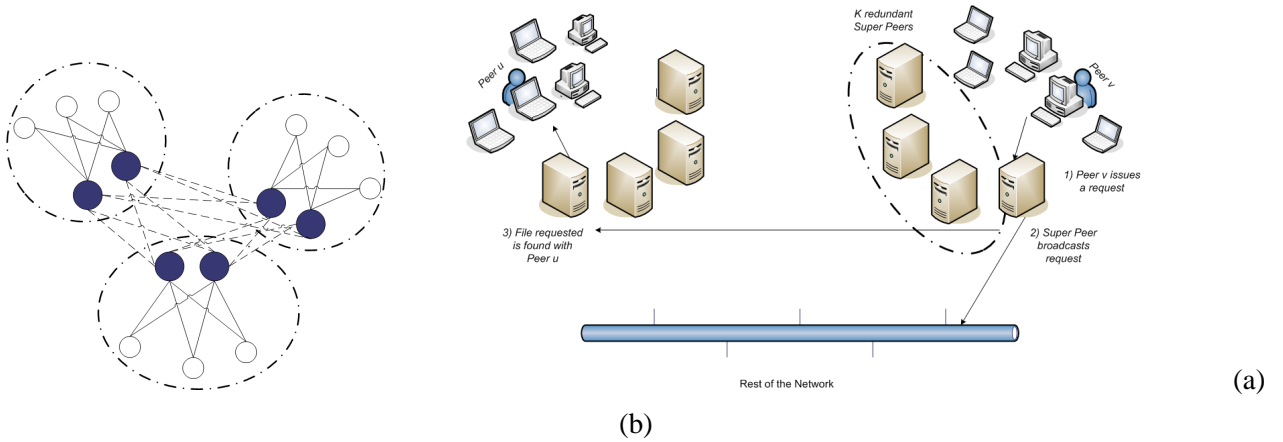


Figure 1: a) Example of a 2-redundant super peer network. b) Request Scenario in SuperTrust

Associated with each peer v is a chosen set of n Super peers (*aggregators*) that are responsible for “collecting” the votes/reports of other peers that have interacted with v and aggregate the results. The aggregators for each peer are chosen amongst the K super peers responsible for the various clusters. Furthermore, in each cluster, a *storage* node is chosen amongst the K super peers as a storage facility for the reputations of the peers/resources located in the corresponding cluster.

SuperTrust takes advantage of the trustworthiness of the Super peers, in order to delegate the responsibility of aggregating/storing of the votes. Furthermore, such a semi-centralized, semi-distributed approach reduces the stress

in the network, thus improving the overall performance and reducing the probability of failures in the system. The fact that super peers are the aggregators/storage peers guarantees that each aggregator/storage peer is within a fixed number of hops from each peer. This should reduce the overall latency incurred in the network. The various actions of a peer v in SuperTrust are broken down into the following steps:

1. *Send a file request:* Peer v issues a request for some resource r (Figure 1(b)). One of the super peers responsible for v 's cluster broadcasts this request to their neighbors.
2. *Receive a list of peers that have the requested file, along with their global rating:* Upon reception of v 's request, each super node checks whether the resource requested is within its cluster. Assuming that the resource is in possession of peer u , the latter issues a reply confirming his possession of the requested resource. In addition, the n aggregators of u respond to v with their decrypted *shares* allowing v to compute the final trust value, as shown in Figure 2(a) and explained further in Section 4.6.

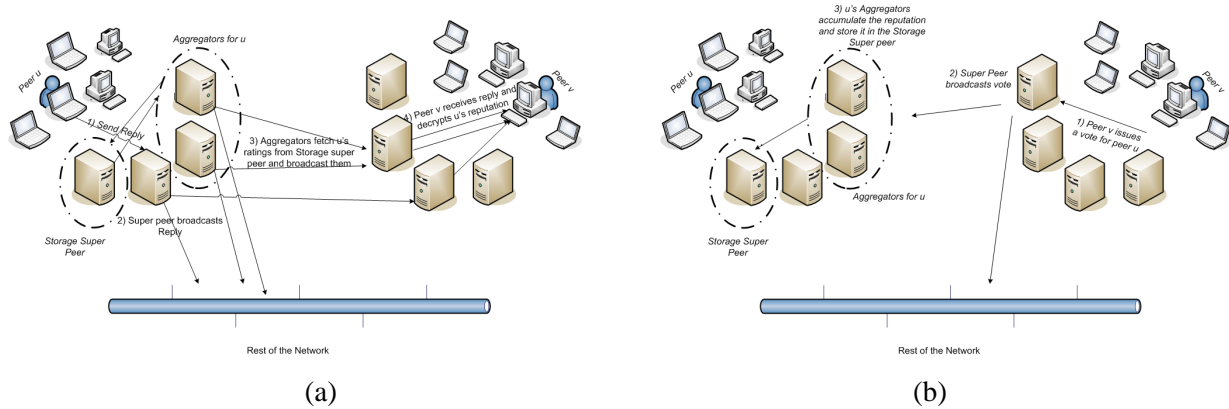


Figure 2: a) Reply Scenario in SuperTrust. b) Vote Scenario in SuperTrust

3. *Select a peer or a set of peers, based on a reputation metric in order to download the resource:* Upon reception of the replies and the decrypted shares from a sufficient number t of aggregators, peer v calculates the global trust value of the replying peers (Section 4.5) and chooses to download the resource from the most reputable peer.
4. *Send Vote:* Then, peer v rates the interaction it had with peer u . It first encrypts the report encapsulating its rating for both peer u and its resource and then submits it to the designated Super peer. The latter forwards the encrypted vote to its neighbors. This process is depicted in Figure 2(b) and explained further in Section 4.4. Upon reception of the encrypted vote, and using appropriate cryptographic schemes, the global trust value of v is updated by v ' aggregators *without* decrypting the intermediate reports, thus ensuring privacy and integrity of votes. At this point, the global trust value remains *encrypted* in the system.

The feature that most distinguishes SuperTrust from other contributions is that trust reports remain encrypted and are never opened during the submission or aggregation processes. As for building reputation, SuperTrust allows the combination of server based and resource based reputation (as suggested in [20]). In the rest of the section, we present the details of our scheme.

4.1 Threat Model and Assumptions

We start by discussing the threat and trust models we expect to encounter in trust handling applications. We consider insider and outsider attacks and mention points of trust in our system.

In an outsider attack, the attacker is not an authorized participant of the network. As peers exchange messages, a passive attacker can easily eavesdrop in an attempt to steal private or sensitive information. The adversary can also alter, spoof or replay messages, trying to create erroneous trust values. The use of proper cryptographic primitives helps defend against these types of attackers.

The existence of inside attackers is more important from a security point of view. These are malicious peers who are authorized participants in the network and come equipped with the right cryptographic material so that they can participate in the various phases of the protocol. Despite this fact, we make sure that privacy and anonymity of benign peers is guaranteed. However, there always exists the possibility that such *malicious peers can rate other peers with poor ratings* in spite of good performance. We don't deal with this type of behavior as this should be tackled by the underlying trust model or architecture.

Our protocol relies on the existence of some authority or mechanism that can generate or certify special purpose keys, assign groups of aggregators, etc. when a peer joins the network. Since we will be relying on the use of public key cryptography, this entity can be viewed as a certifying authority and its public key can be trusted as authentic. Alternatively, this authority can be thought as an entry point in the network, a server that bootstraps the security infrastructure[14]. We will call such a server the *Certification Authority (CA)* and its public key, PK_{CA} , will be known by all peers in the network (this information can be distributed during peer join, for example). We also assume that the underlying Super peer system provides for anonymity of information providers as long as these providers do not reveal themselves any identifying information. Thus we rely on the message forwarding mechanism of the Super peer system to provide for anonymity of broadcasted messages (see [14] for a similar assumption).

Finally, we use trust functions that rely on simple aggregations. While such trust functions can be vulnerable in reputation systems, we assume that these values take into consideration more general transactions contexts, histories, credibility of issuers, etc.[21]

4.2 Definitions and Cryptographic Tools

We are assuming that the reader is familiar with the concept of public key cryptography.

Definition 1 *We say that an encryption function $E()$ is (additive) homomorphic if it allows computations with encrypted values. More precisely, for any two messages M_1 and M_2 encoding integers,*

$$E(M_1) \cdot E(M_2) = E(M_1 + M_2).$$

In a similar manner if the encryptions $E(M_i)$ of messages M_i , $i = 1, 2, \dots, k$, are available then

$$\prod_{i=1}^k E(M_i) = E\left(\sum_{i=1}^k M_i\right).$$

This property is necessary to achieve anonymity as well as verifiability since the aggregated trust value can be computed *without* the decryption of individual reports submitted by interested parties. The decryption is performed only on the *final sum*, guaranteeing privacy of voters.

In [22], Paillier proposed an RSA-type public key cryptosystem based on computations in the group $Z_{N^2}^*$ (the set of numbers relatively prime to N^2), where N is the product of two large primes p and q . This scheme has two very attractive properties: it is homomorphic and allows for efficient decryption (details omitted). However, in order to prevent authorities from learning the contents of the submitted reports and to ensure the privacy of the voters a *threshold* version of the Paillier cryptosystem is needed.

Definition 2 [23] *A (t, n) threshold secret sharing scheme distributes a secret among n participants in such a way that any t of them can recreate the secret, but any $t - 1$ or fewer members gain no information about it. The piece held by each participant is called the share of the secret.*

In our case, instead of having a single authority decrypt the encrypted tally, n authorities share the decryption key, so that at least t are needed to perform the decryption operation. Such versions of the Paillier cryptosystem have been presented in [24]. Below we give a general description of the threshold decryption model. The participants is a set of n players (our aggregators) that share the decryption key and users (peers that submit confidential reports).

- *Initialization*: The players run a key generation protocol to create the public key PK and the secret shares SK_i of the private key SK . This is done securely[25] so that no player can find anything about shares other than its own.
- *Encryption*: The user uses PK to encrypt a message.
- *Decryption*: To decrypt a ciphertext c , each player uses its share SK_i to produce a partial decryption c_i together with a *proof of validity* for the partial decryption. These proofs allow interested parties to verify that the partial decryption is valid even without knowing the underlying secret SK_i . This protects from malicious players submitting erroneous results. The final decryption of c can be produced, if t or more valid decryptions are collected.

We now proceed to discuss the various phases of SuperTrust. The following notation will be used throughout the paper:

Notation	Meaning
$M_1 M_2$	Concatenation of messages M_1 and M_2 .
PK_u, SK_u	Public and private (secret) key of peer u .
$E_u(M)$	Encryption of M , with the public key of u .
$Sig_u(M)$	Signature on M using the private key of u .
T	Timestamp.

4.3 Initial Setup

When a peer v joins the network, the CA assigns a random set of the Super peers pertaining to its cluster to be the n aggregators for v . These will be responsible for decrypting the aggregated trust value of v that remains encrypted in the system. These Super peers may also be aggregators for other peers in the cluster. This group has an associated public/private key pair $\langle PK_A^v, SK_A^v \rangle$. The public key is used by other peers u to rate their interactions with v and submit their encrypted reports into the system. The authenticity of the public key can be verified as it is equipped with a *certificate* carrying the signature of the CA and an indication that this is the key used to submit reports only for v . The decryption key is *shared* among the aggregators using the threshold cryptosystem.

In addition, the CA assigns a storage Super peer for each cluster chosen from the most reputable and trusted Super peers. The storage Super peer serves as a storage repository for the *encrypted* ratings of the peers lying within its cluster. We stress at this point that the storage peer *is not able to decrypt* any rating it possesses; its sole function is to store the rating advertised and accumulated by the various aggregators.

4.4 Submitting a Trust Report

When a peer u has interacted with peer v , it can file a report indicating the rating of this interaction. We assume here that u has acquired the public key PK_A^v through a previous reply received from v . This is needed since before interacting with v it first has to find out how reputable v is (see Section 4.6). Given the public key, u constructs the following message m :

$$m = \text{“Report for } v \text{”} \mid E_A^v(\text{Value}) \mid \langle T_i, u, v, Sig_v(T_i, u, v) \rangle \mid T$$

Then it transmits

$$m \mid \text{Sig}_u(m) \mid PK_u \mid \text{Cert}_u \mid PK_v \mid \text{Cert}_v,$$

where the various parts are described below along with the reason behind their inclusion:

- “Report for v ”: Any textual explanation that this is a report for peer v .
- $E_A^v(\text{Value})$: The *encrypted* trust rating of the interaction. This is encrypted using the public key PK_A^v .
- A *proof of the interaction* with v : This proof of interaction is needed in order to prevent the replay of trust reports. It has the form $\langle T_i, u, v, \text{Sig}_v(T_i, u, v) \rangle$, where T_i is the time the interaction took place.
In general when two peers u and v interact, they exchange such proofs with each other. This makes them eligible to vote and protects them from malicious peers submitting bad reports, even without having interacted with u or v .
- A timestamp T : The use of the timestamp ensures that the report cannot be replayed and resubmitted at a later time by a malicious peer.
- A *signature* $\text{Sig}_u(m)$ of the whole message by u : The signature binds all the parts together so that nobody can alter or replace parts of the message.
- The public key PK_u of u : This is needed to verify the signature of u and it carries along a certificate from the certification server CA .
- The public key PK_v of v : This key was obtained after interacting with v and is included in the message so that during the aggregation process (next section) the proof of interaction $\langle T_i, u, v, \text{Sig}_v(T_i, u, v) \rangle$ can be verified as authentic.

Notice the use of timestamps here. T_i is used to validate the interaction between u and v (this proof of interaction concept was also proposed by [14]). The idea is that any report containing this proof outside some reasonable time frame Δt , must be discarded. However, we have found that the use of T_i alone does not hinder report duplication. A malicious peer u that has interacted with v and hence has a *valid* proof of interaction can actually submit more than one report carrying this proof (the protocol in [14] suffers from this type of behavior). However, this problem can be overcome by queuing the reports in the interval Δt so that all but one carrying the same proof of interaction are discarded by the system. The second timestamp T is needed to prevent malicious peers from replaying valid reports, hence affecting the aggregation process.

4.5 Aggregating Trust Reports

When a report of the previous form is transmitted, it must be stored into the system. Trust reports for peer v get aggregated by its designated aggregators that subsequently submit the updated value to the cluster’s storage Super peer, *replacing* previous reports for v .

The invariant that is maintained at any moment is that the current aggregate value resides encrypted in the system. When a new report for v is issued, the format and the validity of the report is checked in reverse order from the generation process described in Section 4.4. Then using the homomorphic property of the encryption scheme the new aggregated value is computed simply by multiplying the encrypted value in the report with the current aggregate for v .

This approach aims at enforcing the consistency of the aggregation process and protecting from malicious aggregators. In fact, and upon receiving the trust report for v , the aggregators fetch from the storage Super peer v ’s previous ratings, update it using the homomorphic property, and submit the aggregation result back to the storage

Super peer in order to guarantee the durability of the ratings in the system. In turn, the storage Super peer receives the various aggregation results, and *only* stores the encrypted value that was advertised by the majority of the aggregators. Such a scheme protects against up to $n/2$ suspicious aggregators (where n is the total number of aggregators in some cluster) that are trying to cheat the system by submitting erroneous aggregation results. Furthermore, in such a case, an alarm can be triggered notifying about the malicious behavior of these super-peers.

4.6 Reporting back Trust Values

When a peer u issues a request for a resource r , it should wait to receive a reply indicating the availability of the resource in addition to the decrypted aggregators' shares that will allow it to construct the global rating for r and its possessor v .

Peer u first transmits a request message into the system containing the ID of r . Such a query is broadcasted anonymously by the various Super peers in the network using the message forwarding mechanism so that the identity of u is protected.

Upon receiving the request for resource r , the various Super peers identify its availability within their cluster, and transmit a reply message containing the ID v of r 's holder. Then after sniffing the reply message, v 's aggregators automatically access the encrypted aggregated value stored in the storage Super peer (Section 4.5) and produce their partial decryptions. Then they respond back with a message of the form

$$\text{“Partial decryption for } v\text{”} \mid D_i \mid Proof_{Valid}(D_i) \mid PK_A^v \mid Cert_A^v \mid T,$$

where the various parts of this message are explained below:

- An indication that this is a partial decryption message.
- The partial decryption value D_i .
- A *proof of validity* $Proof_{Valid}(D_i)$ that the decryption is correct. Using this proof anybody can verify that the decryption was performed correctly.
- The public key PK_A^v along with a certificate signed by the CA . This key will be used by u to submit a trust report to the system (Section 4.4) after it has interacted with v .
- A timestamp T guaranteeing that the decryption is fresh.

Once peer u collects enough shares (at least t), it can compute the final decrypted result. Then upon satisfaction of the reputation of v , it can decide whether or not to interact with v .

The use of a (t, n) threshold cryptosystem protects against malicious aggregators submitting erroneous decryptions. Additionally, our approach offers protection against replay attacks since the timestamp ensures that the partial decryption is up to date. Finally, for extra protection this message may be encrypted with u 's public key, provided u has included PK_u in its initial query.

4.7 Super Node Selection Process

SuperTrust exploits the advantages of Super peer networks in order to achieve high efficiency while securing the trust handling process. However, several tradeoffs exist in such super peer architectures, especially with respect to the supernode selection process. The supernode selection problem is generally very hard and more complex than some known NP hard problems [26] (such as the Dominating Set problem). This problem also relates to the selection of intermediary nodes in ad hoc wireless networks. The supernode selection process revolves about quickly finding qualified Super nodes to replace defected/unavailable Super peers. Such a process is known to be very hard as it needs to maintain load balancing and system's efficiency.

For example, in Gnutella 0.6 [27], Super nodes are selected mainly based on high bandwidth, processing power and availability. However, Gnutella, as well as most other Super peer applications, does not consider any security requirements for the super node selection process. This obvious lack of security aspects in selection protocols could lead to dangerous consequences in terms of the consistency and availability of the system, especially within the scope of SuperTrust.

Since SuperTrust relies on Aggregators/Super Peers for trust handling purposes, a secure supernode selection process ensuring that the selected super peers are chosen amongst the highly trusted and the least vulnerable nodes is needed. Furthermore, supernode selection should be fast and efficient in order to ensure adequate system performance when subject to rapid joins and leaves.

We propose the following selection criteria: in addition to the basic requirements featured by Gnutella, such as high availability, enough bandwidth and fast processing power, Super nodes are chosen amongst the nodes holding valid certificates, obtained from a trusted *CA* as described in Section 4.1, and having high trust ratings from the other peers in the network. Such a method might efficiently prevent malicious nodes from being upgraded to Super nodes.

Upon selection, the selected Super node replaces its predecessor (aggregator/storage node) and obtains the required information for consistent operation by securely exchanging metadata with the other Super Peers. Furthermore, the *CA* might enforce a re-key, by issuing new certificates and Public Keys to the Super peers in the corresponding cluster, since the Private Key of the defected Super peer might be compromised.

5 Security Analysis

Due to space constraints we highlight the security properties achieved by SuperTrust.

- *Privacy and Anonymity:* The privacy of the peers' reports is guaranteed even if up to $t - 1$ aggregators (where t represents the threshold cryptography) collude with each other. The use of a threshold cryptosystem guarantees that no faulty or malicious aggregator can decrypt the report submitted by a peer. Additionally, SuperTrust uses the homomorphic property of the cryptosystem in order to compute the final tally *without* decrypting individual reports. This strengthens peer's privacy since at no point in the submission process will a report be decrypted. Finally all communications are made through secure channels. An external attacker or eavesdropper can infer no information about the contents of a report since these are sent encrypted using public key cryptography. There is, however, the possibility that a peer v may query for its trust value, before and after, an interaction with peer u , thus recovering the vote of u . This attack applies to *all* systems and can be reduced if the aggregators "batch" or "mix" the votes. While there will be a short delay in developing reputations, in the long run the true behavior of peers will show up.
- *Fairness and Soundness:* A malicious node cannot affect the submission process by submitting invalid reports or by not following the protocol. If a peer tries to submit an invalid report or if there is no associated proof of interaction, the report will be discarded by the system. This prevents malicious peers from rating other peers when they have no such right. Notice, however, that two malicious cooperating peers may interact with each other (thus having valid proofs of interaction) in order to elevate their own ratings. We don't deal with this type of behavior in this work as this should be handled by the underlying trust model. Finally, the fairness and soundness of the submission process is also guaranteed against colluding aggregators, as long as no more than t of them cooperate.
- *Persistence:* Persistence of submitted reports is guaranteed by the character of the aggregation process. When a peer submits a trust report, its contents are securely stored in the designated Storage super peer. If this peer ever leaves the system, its role will be assigned to a new super peer, thus guaranteeing maintenance of votes.

- *Eligibility and Unreusability*: SuperTrust ensures that only eligible peers are allowed to cast a report. This is achieved by the incorporation of proofs of interaction in the reports. The inclusion of timestamps in the reports and the proofs of interaction guarantee that reports can be submitted only once, thus preventing report duplication.
- *Verifiability*: In SuperTrust, the nature of the aggregation process and the use of majority by the storage Super peer guarantees that all reports are taken into consideration when computing a peer’s global trust value. Additionally, the aggregators may check the Storage peer for *consistency* since after performing the homomorphic multiplication they can test whether the storage peer has updated the global trust value with the values they submitted.

6 Implementation and Simulation Results

This section presents our implementation setup of SuperTrust under a Super peer network environment. We have implemented SuperTrust in Java. Our implementation is multithreaded and relies on Jade agents [28] as the main infrastructure for exchanging messages between the various peers. Our framework was evaluated on a moderately connected network consisting of various networked computers connected to a 100 Mbit LAN where up to two different processes reside on each machine. We have simulated a total of 1200 peers, 2400 different resources and at least 18 superpeers/aggregators pertaining to six different clusters of 200 peers each. At the beginning of the simulations, each peer has a number of random resources. Furthermore, resources are replicated randomly on the network with a replication index of 1.43. Due to the sufficient numbers of resources and peers in our network, we can fairly evaluate the performance metrics in SuperTrust, based on a realistic scenario featuring adequate resource replication.

In an attempt to take account of the various propagation delays induced in deployed Super peer networks, our simulations feature a round trip time (RTT) value of 195.4 ms (*as advocated* in the Internet Traffic Report) between pairs of super peers and an RTT value of 25.4 ms within the clusters (i.e. between SNs and ONs).

Throughout our simulations, we keep track of the propagation of all of the messages (requests, replies and votes) sent and received. However, depending on the graph’s connectivity, messages can propagate within a loop. We use the following method to reduce the effect of such behavior: each super peer will broadcast a message to all its neighbors except the neighbor that sent him the message in question. Furthermore, each super peer, and upon sniffing a received message, checks and updates a flag indicating that this message was processed by it, therefore guaranteeing that each message is analyzed only once per super peer.

To evaluate the effectiveness of our framework, we conducted our experiments aiming at analyzing three main performance metrics: response time, messaging costs and the effect of malicious aggregators. These scenarios were run in two modes: SuperTrust mode and no reputation mode, the latter representing a non-reputation based Super peer network. In what follows, we present an analysis of our simulations results.

6.1 Messaging Costs

We proceed to evaluate the messaging costs induced in our framework in relation to the total number of requests issued. In this setting, we have evaluated three models: SuperTrust with threshold cryptography of 3 and 5 aggregators, and no reputation super peer model. Figure 3(a) illustrates the obtained results. Each data point in this figure is averaged over 6 times the total number of requests. These results are due to the fact that each issued request triggers at least 5 to 7 additional messages, per reply, in the threshold cryptography cases of 3 and 5 respectively. These numbers pertain to the fact that upon request, at least one peer will reply, followed by the aggregators replies, thus allowing the requestor peer to decrypt the corresponding reputation and send back its own rating for this interaction.

Such messaging overhead can be tolerated even in highly congested networks, when compared to the overhead induced in other proposed protocols. Figure 3(b) shows a comparison between SuperTrust, TrustME[14] and

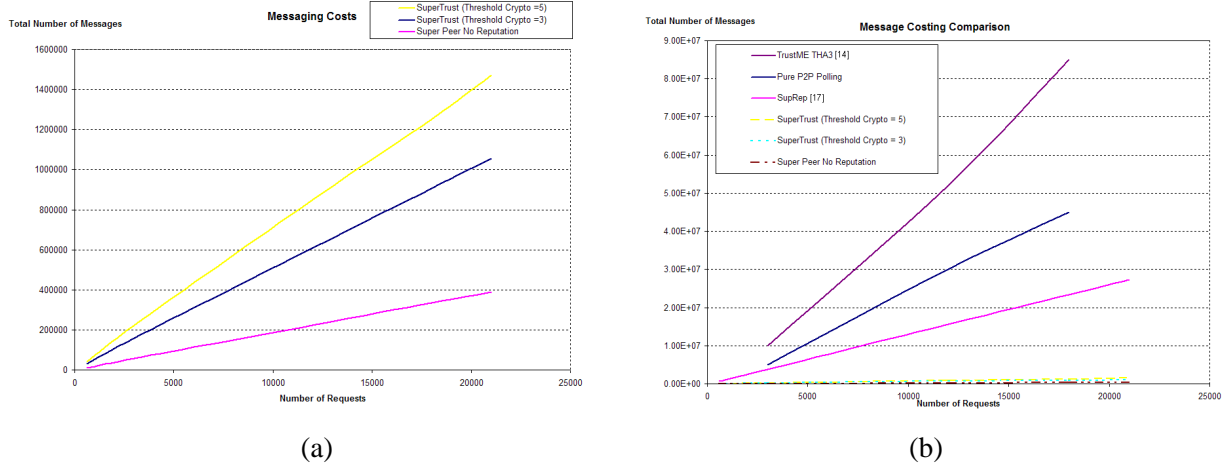


Figure 3: a) Messaging costs in SuperTrust. b) Comparison of messaging costs

SupRep[17], with respect to messaging costs. As it can be seen, SuperTrust achieves much lower overhead when compared to these aforementioned protocols. The main advantage of SuperTrust over TrustME is based on the use of the Super peer model. While TrustME was designed for pure P2P networks, SuperTrust guarantees that each aggregator/storage peer is within a fixed number of hops from each peer. As a result, the amount of traffic in the network is reduced when compared to the pure flooding model. On the other hand, SupRep, established within a super peer environment, exhibits high stress in the network, since it requires the requestor peer to poll all other peers. Such a requirement corresponds to the generation of more than 1200 messages per request, under our settings.

6.2 Response Time

By relying on a real network and by making use of appropriate RTT values, we can correctly evaluate the response time in SuperTrust. We model response time by the time elapsed from sending the request till the time the requestor peer is able to fully decrypt the global rating of the resource holder. In other words, response time corresponds to the time at which enough valid aggregators' decrypted shares are received by a requestor, with respect to the request sending time. We have measured the average response time in SuperTrust in relation to the total number of requests. Our measurements were done for various number of requests, sent at the fixed rate of 15 per second per cluster (90 requests per second in the network), for the threshold cryptography cases of 3 and 5. Figure 4(a) depicts our findings. Every data point in this figure is averaged over almost 10000 runs, in order to ensure consistent measurements.

For these results we considered the time to encrypt/decrypt a vote plus the time to verify the proofs of validity submitted by the various aggregators. For the actual numbers we incorporated the findings of [24]. In this paper it is mentioned that the time for encryption (decryption) is approximately 0.3ms (0.18ms) *per bit* of plaintext message on a Pentium 2.4GHz. Assuming that the user is selecting a value between 1 and 1000 to rate the interaction, we see that the time for encryption/decryption is negligible. Perhaps more important is the overhead due to threshold version of the Paillier cryptosystem. In the same paper, we see that it takes 2 modular exponentiations per share for the proofs of validity (essentially zero-knowledge proofs) plus one more for share combining. Thus it takes 3 modular exponentiations per share to get the final result or $3t$ exponentiations overall for a (t, n) threshold cryptosystem.

The results show an advantage of 500ms, on average, per request for the no reputation scheme over SuperTrust. This is mainly due to the overhead induced by the use of threshold cryptography. In fact, experimental setups suggest that a modular exponentiation consumes almost 70 ms for a 1024 bit exponent. As a result, the overall cryptographic overhead induced in SuperTrust with threshold cryptography of 3 can be approximated by 400ms and increases to 600ms for the 5-threshold cryptography case. Notice, however, that the overhead of public key encryption can be cut down 5-6 times if one uses the more efficient *elliptic curve* cryptography. However, we leave this for future

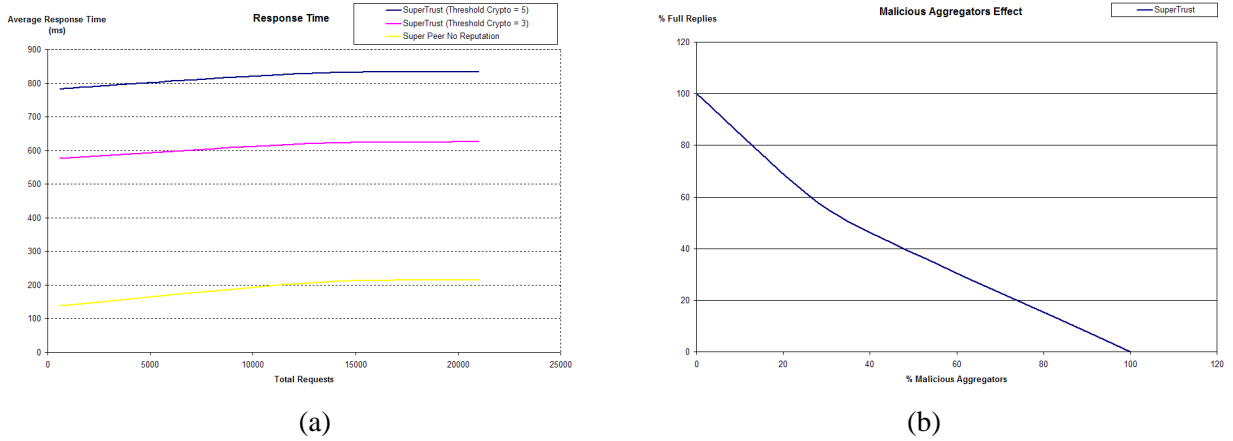


Figure 4: a) Response time in SuperTrust. b) Effect of Malicious Aggregators

consideration. The graph also reveals a slight increase in response time as the total number of requests increase. This is due to the actual state of the network; as requests increase, more requests will be buffered for processing, thus resulting in an increase in response time.

Our investigation has shown that ours is the most efficient system to date since almost all contributions in this area neglect to provide response time. For example, although SupRep[17] is expected to have less cryptographic overhead, since it does not make use of threshold cryptography, it induces more stress in the network because of its requirement to poll for *all* available opinions.

6.3 Effect of Malicious Aggregators

In SuperTrust, the requestor peer must have a sufficient number of valid partial decryptions from some aggregators, in order to correctly retrieve the global rating of another peer. Although malicious aggregators cannot cheat the system by inserting invalid votes, they might not allow a correct decryption of a global rating if they send invalid decryption shares. Fortunately, the use of threshold cryptography helps to minimize such a malicious behavior.

In order to study the effect of malicious aggregators in SuperTrust, we simulated a 6-redundant Super peer network featuring 5 aggregators, and a cryptographic threshold of 3. In this setup, requests were issued at a rate of 90 per second for a total of 18000 requests. In our model, malicious aggregators do not send their shares at all. This can be seen as equivalent to the case where the suspicious aggregators broadcast invalid decryption shares since in both scenarios, such a behavior might hinder a correct decryption of the global ratings. In Figure 4(b), we plot the percentage of malicious aggregators versus the percentage of the global ratings that were successfully decrypted. Each data point is averaged over 6 runs. Moreover, 100% malicious aggregators in our setting, is equivalent to having 3 or more bad aggregators within each cluster, since, under such situations, each peer will receive 2 partial decryptions at most (while the cryptographic threshold for complete successful decryption is 3 in this setting).

Our simulation results show that, in spite of the presence of 30% malicious aggregators in the network, almost 60% of the requests will have sufficient partial decryption from the aggregators in order for the requestor peer to fully decrypt the advertised rating. As a result, SuperTrust proved its ability to sustain malicious behaviors by removing single points of failure through the use of efficient threshold cryptography techniques.

7 Conclusions and Future Research

In this paper, we have presented SuperTrust (Super peer Trust Handling), a framework that manages trust ratings of peers in a way that preserves the privacy and anonymity of transactions, in the context of Super peer networks.

SuperTrust achieves these properties by ensuring that trust reports remain encrypted and are never opened during the submission or aggregation process. The use of threshold cryptography allows the aggregators to have access to such ratings and compute a global trust value without the need to see the individual reports. Each aggregator produces a partial value, and when enough of these shares are collected, the final trust value can be computed in a simple and mechanical way. Thus, SuperTrust is resistant to attacks by colluding Super peers. Additionally, it offers properties like persistence, eligibility and unreusability of reports, similar in some sense to the characteristics of electronic voting systems. Furthermore, SuperTrust exploits the benefits of Super peer networks to reduce messaging overhead in the network, while providing robust and efficient security. We feel that SuperTrust, along with this first precise definition of properties that *any* protocol for handling trust values must satisfy, will demonstrate the usefulness and viability of our approach and stimulate further research in this area.

We currently investigate the possibility of eliminating the use of public key cryptography (PKC) or using elliptic curve variants that will improve response time considerably. In this work, we use PKC to encrypt reports and aggregate them in a secure and efficient way, however we are also studying the possibility of maintaining shared trust reports using proactive public key cryptography or using lighter symmetric cryptography.

To enhance the verifiability property of the system we also consider a *bulletin* based approach, where anyone can verify the result of the submission process and test its correctness. All reports will be posted in a bulletin board and will remain available for future consideration so that anybody can verify their validity. Furthermore, the partial decryption results produced by the aggregators can also be published in the bulletin board, along with their proofs of validity. This way *any* external party could verify the proofs and be convinced that the final result is correct.

References

- [1] KaZaA: Available from <http://www.kazaa.com/us/index.htm>.
- [2] The Gnutella Protocol Specifications v0.4. Document Revision 1.2. Available from <http://www.clip2.com>.
- [3] I. Clarke, O. Sandberg, B. Wiley, and T. Hong, "Freenet: a distributed anonymous information storage and retrieval system", In *Proc. of the Workshop on Design Issues in Anonymity and Unobservability (Berkeley, CA)*, pp. 311–320, July 2000.
- [4] S. Marti and H. Garcia-Molina, "Identity Crisis: Anonymity vs. Reputation in P2P Systems", In *IEEE 3rd International Conference on Peer-to-Peer Computing*, P2P 2003.
- [5] eBay, Available from <http://www.ebay.com>.
- [6] D. Dutta, A. Goel, R. Govindan, and H. Zhang, "The Design of a Distributed Rating Scheme for Peer-toPeer Systems," In *Workshop on Economics of Peer-to-Peer Systems*, 2003.
- [7] K. Aberer and Z. Despotovic, "Managing trust in a peer-to-peer information system," In *10th International Conference on Information and Knowledge Management*, 2001.
- [8] S. Kamvar, M. Schlosser, and H. Garcia-Molina, "Eigenrep: Reputation management in p2p networks," In *12th International World Wide Web Conference*, 2003.
- [9] Yao Wand and Julita Vassileva, "Trust and Reputation model in Peer-to-peer Networks", In *3rd IEEE International Conference on Peer-to-Peer Computing*, 2003.
- [10] L. Xiong and L. Liu, "A reputation-based trust model for peer-to-peer e-commerce communities," In *IEEE International Conference on Electronic Commerce*, 2003.

- [11] Nikos Ntarmos, Peter Triantafillou, “SeAl: Managing Accesses and Data in Peer-to-Peer Sharing Networks,” In *4th IEEE Int. Conference in Peer-to-Peer Computing*, pages 116–123, 2004.
- [12] O. Baudron, P.-A. Fouque, D. Pointcheval, G. Poupard, and J. Stern, “Practical Multi-Candidate Election System,” In *Proc. of the ACM Conference on Principles on Distributed Computing*, August 2001, Philadelphia, USA
- [13] J. R. Douceur, “The sybil attack,” In *Proc. of the 1st Inter. Workshop on Peer-to-Peer Systems*, Cambridge, MA (USA), 2002.
- [14] Aameek Singh and Ling Liu, “TrustMe: Anonymous Management of Trust Relationships in Decentralized P2P Systems,” In *3rd International IEEE Conference on Peer-to-Peer Computing*, 2003.
- [15] L. Xiong and L. Liu, “PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities”, In *IEEE Transactions on Knowledge and Data Engineering*, Vol. 16, No. 7, July 2004.
- [16] F. Cornelli, E. Damiani, S.D.C. di Vimercati, S. Paraboschi, and P. Samarati, “Choosing Reputable Servents in a P2P Network,” In *11th International World Wide Web Conference*, 2002.
- [17] S. Chhabra, E. Damiani, S. De Capitani di Vimercati, S. Paraboschi and P. Samarati, “A Protocol for Reputation Management in Super-Peer Networks”, In *Proc. of the 15th Inter. Workshop on Database and Expert Systems Applications,(DEXA04)*.
- [18] L. Mekouar, Y. Iraqi and R. Boutaba, “Peer to Peer’s most wanted: Malicious Peers”, In *The International Journal of Computer and Telecommunications Networking archive Volume 50, Issue 4*, (March 2006).
- [19] B. Yang and H. Garcia Molina, “Designing a Super-Peer Network”, In *Proceedings of the ICDE*, March 2003.
- [20] E. Damiani, S. De Capitani di Vimercati and S. Paraboschi, “A Reputation Based Approach for Choosing Reliable Resources in Peer to Peer Networks”, *CCS’02 Nov 18-22, Washington DC, USA*, 2002.
- [21] A. Jøsang, R. Ismail, C. Boyd, “A Survey of Trust and Reputation Systems for Online Service Provision”, In *Decision Support Systems*, 2006.
- [22] P. Paillier, “Public-Key Cryptosystems Based on Discrete Logarithm Residues,” In *Eurocrypt ’99*, LNCS 1592, Springer-Verlag.
- [23] A. Shamir, “How to share a secret,” In *Comm. of the ACM*, 22:612–613, November 1979.
- [24] I. Damgård and M. Jurik, “A Generalization, a Simplification and some Applications of Paillier’s Probabilistic Public-Key System,” In *PKC ’01*, LNCS 1992, pages 119-136, Springer-Verlag, 2001.
- [25] I. Damgård and M. Kopolowski, “Practical Threshold RSA Signatures Without a Trusted Dealer,” In *Eurocrypt ’01*, LNCS, Springer - Verlag, 2001.
- [26] V. Lo, D. Zhou, Y. Liu, C. GauthierDickey, J. Li, ”Scalable Supernode Selection in Peer-to-Peer Overlay Networks” In *Hot Topics in Peer-to-Peer Systems*, 2005
- [27] The Gnutella Protocol Specifications *Version 0.6*.
- [28] Jade: Java Agent DEvelopment Framework, Available from <http://jade.tilab.com/>.