X^2Rep: Enhanced Trust Semantics for the XRep Protocol

Nathan Curtis, Rei Safavi-Naini, and Willy Susilo

Centre for Information Security Research
School of Information Technology and Computer Science
University of Wollongong
Wollongong 2522, Australia
{nathanc,rei,wsusilo}@uow.edu.au

Abstract. Peer-to-peer file sharing networks are a popular means of sharing a diverse range of resources and information. Many of today's most widely used file sharing networks are built on the Gnutella file sharing protocol. The open, insecure nature of such networks means that they are susceptible to the distribution of malicious, unauthentic or low quality resources. XRep is a reputation-based trust management system designed to reduce the number of malicious or low quality resources distributed in a Gnutella file sharing network. XRep is significant in that it can be integrated into a Gnutella environment with minimal disruption. This is achieved primarily through the use of the same message passing mechanism as in the standard Gnutella protocol. We demonstrate that the trust semantics algorithm employed by XRep has a number of weaknesses and does not produce correct trust values when used against a range of strategies that can be employed by malicious agents. We describe an enhanced trust semantics algorithm called X^2Rep that can be seamlessly incorporated into the XRep protocol. We demonstrate that this algorithm is robust against such strategies, offers a high degree of expressiveness in voting and vote evaluation and significantly reduces the network communications required by the XRep protocol.

1 Introduction

Peer-to-peer (P2P) file sharing networks have become a popular way of distributing a diverse range of resources and information. P2P systems are truly decentralized systems that are believed to reflect society better than other types of computer architectures. In a P2P network each node is a client and server both, and by participating in the network allows others to access its computing resources. P2P networks have a number of attractive properties including scalability, anonymity and fault-tolerance, that are much harder to achieve in traditional networks. Nodes can join and leave the network without leaving any trace and while active can initiate downloads and respond to queries. However due to the lack of accountability, such networks have tremendous potential to be misused. For example a malicious peer can use the network to distribute malicious code [7].

Another important problem is *authenticity* and *quality* of downloaded resources. Unauthentic or poor quality resources could be deliberately shared by the casual user. A peer that has requested a resource may receive one or more response(s) and needs to decide which one, if any, to download. In the absence of any mechanism to differentiate between good and poor quality resources a peer may have to download a resource many times and this will result not only to the high network cost but also contribute to network load and slower downloads.

Traditional methods of providing security in networks cannot be implemented effectively as the heavy use of cryptography will not only slow down the network but also may be unacceptable to users with less powerful computers. An approach to increase reliability of P2P networks without loosing their essential properties including anonymity is to use *reputation* systems to identify the quality of peers and resources. A reputation system collects, processes and distributes information about entities based on their history in the system [7]. For example in a P2P system, a peer's reputation may be determined by its behaviour in previous transactions, and a resource reputation may be determined by the evaluation of peers who have downloaded the resource.

In [5], a reputation based trust management system for the Gnutella protocol was proposed that has a number of attractive properties. The system uses reputation of peers and resources both, to assist a requesting peer in selecting which resource to download. The reputations generated by the system allow a user to have an indication of the level of risk associated with the download, hence enabling him to make the required provisions. This is the first system that includes reputations of resources and is shown that because of this inclusion a number of known attacks can be prevented. An important feature of the system is that the reputation system can be incorporated into the Gnutella protocol and the additional information be piggybacked onto the existing Gnutella protocol.

1.1 Our Contribution

We present a trust semantics algorithm called X^{2}Rep that extends the XRep protocol. The purpose of X^{2}Rep is to address the weaknesses of XRep. We demonstrate that our algorithm provides substantial improvements against these weaknesses using extensive simulations. We give more expressive power to peers to express their opinion about resources that they have downloaded and the peers that they have downloaded from. We allow collusions of malicious peers to use a range of strategies and use the reputation to protect against these attacks.

A major challenge to the development of a reputation system is to ensure the reliability of gathered reputation information. In particular, it is vital that any "vote spoofing" activity is as difficult or expensive as possible for malicious agents. The XRep protocol uses a complex process of challenge and response messages to ensure that a vote is supplied by a 'real' peer. We eliminate this complexity by employing extensive vote generation and evaluation system that makes use of voter *credibility* information. Voter credibility is an additional piece of information that helps an evaluating peer to determine the trustworthiness of a voter's vote through the evaluation of the voter's previous voting activity.

1.2 Related Work

Reputation-based trust management systems must address issues at two levels [2]: 1) Data Management, and 2) Trust semantics. Data management is concerned with the storage and dissemination of reputation information in a distributed environment with no centralised control. Trust semantics specify the model for the evaluation of 'trust' through the computation of gathered reputation information.

Data management techniques used in distributed reputation-based trust management systems fall into two broad categories:

1. Peers maintain repositories of their experiences and make it available to others through a voting mechanism;
2. Reputation information is held in the network and is accessed through an additional network overlay, such as a distributed hash table (DHT).

Work in the former category includes XRep [5] and its predecessor, P2PRep [4]. In both protocols the reputation information is piggybacked onto the Gnutella P2P file sharing protocol. In the P2PRep protocol reputation information is associated only with peers.

Work in the latter category includes EigenRep [6] that uses a distributed hash table as its network overlay. Another system in this category is proposed by Aberer and Despotovic [2] and uses a P-Grid [1] as its network overlay. A novel aspect of this system is the use of a complaint system for assigning reputations.

The rest of this paper is organised as follows. In Section 2.1 we give a brief overview of the Gnutella protocol and XRep protocol. Section 3 gives our analysis of the system and its shortcomings. Section 4 defines the properties that must be found in a reputation system. Section 5 describes X^2Rep, our trust semantics algorithm. Finally, Section 6 concludes the paper.

2 Peer-to-Peer File Sharing

Recent years have seen a tremendous growth in the popularity of peer-to-peer (P2P) file-sharing networks [9]. Traditionally, the term P2P has been used to describe a decentralised network architecture in which all peers have equal roles and responsibilities, and follow the same behavioural patterns. In a P2P network, a peer acts as both client and server and exchanges information and services directly with other peers. Often, a peer also acts as a router, forwarding messages it receives to directly connected neighbours.

Each peer in a P2P file-sharing network participates by offering files for downloading by other peers. A file exchange interaction follows two phases; a *search phase* in which the enquirer attempts to locate a peer offering the desired file, and a *download phase* in which the peer connects directly with the offerer to initiate the download, commonly using traditional protocols such as HTTP or FTP. Many of todays most widely used P2P file sharing applications are based on the Gnutella protocol [8].

2.1 XRep Protocol

XRep [5] is a notable reputation based trust management system that can be straightforwardly piggybacked onto the Gnutella P2P file sharing protocol. XRep defines a secure protocol for the exchange of reputation information using the same message passing mechanisms as used in standard Gnutella Query andQueryHit exchanges. Thus, to provide XRep functionality, current Gnutella implementations require only modest modifications.

In XRep reputation information is associated with both peers and resources. XRep requires resources and peers to be uniquely identifiable. This is achieved by using the digest of a resource's content as the $resource_{id}$, and the digest of the public key of a peer as the $peer_{id}$. Using a cryptographic hash function ensures that the resources and the peers are uniquely identifiable.

When considering a file download in Gnutella, the user selects the resource that best satisfies the request (using information such as the standard resource meta data string and offerers connection speed). To assist the user in making the download decision, the network is 'polled' for any available reputation information on that resource and the peers that offer it. Poll messages are broadcast in the same way as Gnutella Query messages. All peers maintain repositories of their experiences (both good and bad) of resources they have downloaded and the peers with whom they have interacted. When a peer receives a Poll message, it checks its repositories for matching resource and peer identifiers. If it has some information to offer, it generates a set of binary votes based on its experiences, and returns them to the enquirer as a PollReply message.

The resource and peer votes are then processed and combined to produce a single value to the user as a reputation value for the download under consideration. Based on this reputation value, the user can make a decision whether or not to initiate a download.

Prior to the download, the offering peer for whom the highest peer reputation value was calculated is contacted directly to verify that it has really offered the target resource. This exchange is known as the Best Peer Check.

We note the following about the protocol.

Phase 1. A minor change to the Gnutella Query exchange is required; the resource identifier is added to the resource information contained in the ResultSet of the QueryHit message. This allows the polling peer to uniquely identify each offered resource.

Phase 2. The poll message consists of the identifier of the resource under consideration and the set of peers that offer it. Also included is a public key Pk_{poll} for which only the polling peer knows the private key. This may be a persistent key pair or a pair generated on the fly for each poll. Voting peers return their votes for some or all of the entities listed in the Poll message together with their IP address. The message is encrypted with Pk_{poll} to ensure confidentiality.

Phase 3. Once a set of votes are received, the polling peer must try to ensure the reliability of the votes and the honesty of the voters. The polling peer attempts this by carrying out the following steps.

– Decrypt each PollReply message and detect any tampering that may have taken place.
– Group votes from voters that are from the same IP network.
– Select a portion of peers from each group send a *TrueVote* challenge, from which the poller expects to receive a *TrueVoteReply*. This ensures that at least some of the votes are from genuine peers and not merely spoofed votes from non-existent IP addresses.

Phase 4. At this stage the polling peer has evaluated trust for all the entities under consideration. The poller now carries out one further phase to ensure that the peer with the best trust evaluation exists and actually offers the resource. It is important for two reasons:

– A malicious peer is prevented from 'hijacking' the identity ($peer_{id}$) of a reputable peer.
– If it can be established that the resource has a good reputation and is offered by a peer with a good reputation, then it is possible to download that resource from any offerer and be assured that the resource is reliable. This can be considered as a load balancing technique.

3 Evaluating XRep

XRep uses the same constrained broadcast and back propagation mechanisms as used in the standard Gnutella Query and QueryHit exchange and therefore effectively doubles the amount of traffic required to complete a single transaction. A number of additional messages must also be exchanged to ensure vote reliability and the existence of voting peers.

The main shortcoming of XRep is the inadequacy of trust semantic and calculation of reputation values. In XRep a peer's experience repository consists of a table that contains a binary value for each resource describing the peer's opinion, good (+) or bad (-), about the resource, and a peer repository, which includes triplets of ($peer_{id}$, num_{plus}, num_{minus}) that records the number of good and bad download counts for each peer.

When polled, a peer converts these experiences into a binary vote for each entity matched in the poll message. Although these values are adequate to provide rudimentary information on whether a peer or resource is good or bad, finer evaluations such as the voter's judgement on the quality of a resource cannot be expressed. This results in the reputation calculation becoming ineffective against a range of malicious strategies. Important successful malicious strategies are the following.

– The generation of "spoofed" positive votes from fake peer identities.
– The systematic generation of positive votes for other members of a voting clique.
– The generation of negative votes for genuine peers in order to reduce their evaluated trust value.

The XRep protocol attempts to ensure the reliability of votes and protect against votes originating from colluding peers. This is by identifying voting cliques through clustering the votes that are provided by voters with the same network portion of their IP address. Such a correlation between colluding peers and IP addresses is tenuous because,

– Users connecting via a proxy server will share the same network part of their IP address and will therefore be considered as part of a voting collusion. It is therefore likely that a substantial number of legitimate votes will be treated as malicious.
– It is highly likely that, in the real world, malicious agents will have completely different IP addresses, for example, if they subscribe to different providers. These agents will therefore be able to continue generating spurious votes unchallenged.
– The protocol requires that a portion of the clustered peers be directly contacted to ensure that the they have actually voted. It is impractical to directly contact any more than a very small proportion of peers from each cluster and therefore a large amount of spurious voting activity could potentially continue unchallenged.

XRep provides some safeguards against ID Stealth attacks. These attacks take place when a malicious peer 'hijacks' the identity ($peer_{id}$) of a reputable peer in order to deceive another peer into a malicious download. In such cases, the downloading peer believes it is interacting a peer with a good reputation. XRep provides safeguards against this attack in the Best Peer Check message exchange. Prior to downloading a resource, the downloading peer challenges the offering peer as to whether it really does offer the resource under consideration. The offering peer sends a response that is signed using its private key, and also supplies its public key. The downloading peer can be certain of the identity of the offering peer, firstly by verifying the signature of the message, and secondly by taking a cryptographic hash of the provided public key and comparing it against the $peer_{id}$ of the offering peer. If all verification is successful the downloading peer can initiate the download.

3.1 Malicious Strategies

We focus on three basic strategies that can be employed by a single malicious peer or a group (collusion) of malicious peers with the intention of circumventing or degrading the reputation system in order to continue to share malicious resources unchallenged. We outline these strategies in the following sections.

Strategy A. This strategy is the simplest way for a malicious peer to share malicious resources. The peer actively participates in the network by offering good resources. However occasionally the malicious peer will offer malicious resources. The malicious peer must carefully monitor the amount of good and bad resources it supplies in order to maintain a network-wide reputation that is sufficiently high for other peers to deem it trustable.

Strategy B. In this strategy a malicious agent attempts to degrade the quality of the reputation system by generating spurious votes when polled. The principal objective of this strategy may either be to simply degrade the correctness of reputation values to the point where these information are no longer trustable, or to attempt to increase the peer's relative standing by voting positively for itself and negatively for all others.

Strategy C. This strategy shares a similar objective with Strategy B. The principal differentiator is that more effort and resources are required on the part of the malicious peer(s) and such activity is harder to counteract by the reputation system. A group of peers systematically vote positively for each other whilst sharing malicious resources. Each peer in the group may also share some good resources in order to enhance its own reputation. The difficultly in detection of this strategy results from the evaluating peer receiving what appears to be a set of valid votes sent by real peers.

Other strategies hybrids of the basic strategies identified above to further increase their effectiveness.

4 Reputation-Based Trust Management for Peer to Peer Networks

The aim of a reputation system is to provide some kind of 'rating' that can be used by users to select a resource and a peer from which the resource will be downloaded. The reputation system, at each time t, will result in a number $\sigma^x(t)$, the reputation score for x, such that a high score represents the genuineness of x (peer or resource) and low score, shows the opposite, and t is the time. A reputation system must satisfy the following properties.

1. Correctness and Soundness: the system must ensure that genuine entities x will eventually receive high σ_0^x and fake entities (malicious peer or resource) will eventually receive low σ_0^x, where σ_0^x denotes a *True Score*.

2. Dynamic behaviour: Reputation scores vary over time. We require that the reputation of an entity x to *stabilize* to a *True Score*, σ_0^x. That is although the instant value of reputations will change but after a transient phase their values will be within an ϵ error from the True Score. We are also interested in the transient behaviour of the function $\sigma^x(t)$ with time: that is the rate of convergence of $\sigma^x(t)$ to σ_0^x. This is a measure of effectiveness of the system.

There are a number of conflicting requirements on the dynamic behaviour of the system.

1. *Start up*: A reputation system must provide a strategy for a genuine entity (resource or peer) to join the system (resource to be chosen for download, and peer be selected to download from). However the start-up strategy must prevent malicious agents from entering the system.
2. *Runtime protection*: A reputation system must ensure that an existing malicious entity will loose its reputation, even if it starts with high reputation value, after a defined length of time.

The strategies that are used to speed up Runtime behaviour in general will make the start up phase of genuine entities slower. Balancing the two requirements must be done for each particular system.

$\sigma^x(t)$ converges to σ_0^x in minimum time. We accept that since reputation is a function of time over a period of time, its value could be different from its true value. We require that the system can be stabilized to its true value.

4.1 Requirements of a Reputation System

Reputation values are evaluated through the past experience of a peer and its response to a current query. Applying the above principles to an XRep type reputation system for P2P file sharing we will have the following groups of requirements.

Security Requirements

S1 Honest peers should be able to join the network and introduce resources in the network as they wish. The reputation of both resource and peer should raise to a level τ, determined by the network designer, that gives them a reasonable chance of being chosen by other peers.

S2 Reputations must be calculated and and propagated through the network securely, and at a sufficient rate to allow timely identification of malicious resources and/or peers.

We note that requirement S2 implies that a secure communication environment exists. However following XRep approach we will not assume a secure communication layer and will incorporate the required security as part of the reputation system. This includes protection against the deletion of and tampering with recommendations whilst in transit. We note that we need not consider source authentication because of the anonymous nature of the system.

One may assume such a layer to be able to focus on the design and behaviour of reputation functions and their dynamics.

Operational Requirements

1. A reputation system should maintain the essential properties of the underlying P2P system. In the case of Gnutella, this includes a decentralised architecture, network transitivity, and anonymity of participants.
2. Reputation system can assist with balancing the load in the system. That is, use reputation values as a mechanisms to ensure avoidance of unnecessary bottlenecks by using harsh assessments of entities (for example by using only zero and one for reputation values).

A reputation system will add some communication and computation cost to the original system.

Efficiency

1. The reputation system must not produce excessive network traffic to the extent that the service provided by the underlying network is degraded.

2. A reputation system must not require excessive storage space and computation power from peers in the network.

In the following section we propose modifications to XRep protocol to provide protection against the malicious strategies outlined in Section 3. We describe how this can be achieved at little cost to each peer, requiring only a modest amount of additional storage space and computational power. We also describe how the introduction of voter credibility allows XRep network communications to be simplified, reducing network traffic.

5 X^2Rep

X^2Rep is designed to address the weaknesses of the XRep protocol. The X^2Rep reputation system provides safeguards against threats posed by collusions of malicious peers, attempting to circumvent the system and causing malicious downloads. The algorithm achieves its security goal whilst reducing communications overhead. This is achieved by determining the trustworthiness of voters by using voter credibility, rather than by clustering voters and requiring that a portion of them confirm their vote.

5.1 X^2Rep Trust Semantics Algorithm

We describe the system by breaking it down into four logical parts: 1) Local Reputation Repository; 2) Voting; 3) Evaluating Ratings for Downloads; and 4) Updating State on a Peer. Each of these is described in detail in the following sections.

Local Reputation Repository. Each peer will store data expressing its experiences with peers and resources that it has interacted with. For each downloaded resource with identification string $Resource_{id}$, it stores a pair ($Resource_{id}$, $\lambda_{Resource_{id}}$), where $\lambda_{Resource_{id}}$ is a real value between 0 (poor or malicious) and 1 (good), that is a measure of satisfaction of the peer with the resource.

For each peer P_j that P_i has interacted with, P_i maintains a vector of length n storing its past n experiences with that peer . The peer Experience Vector υ_{ij} is denoted by $\upsilon_{ij} = (P_j, (q_{ij,1}, q_{ij,2}, q_{ij,3} ... q_{ij,n}))$ where $q_{ij,k}, k = 1, \cdots n$ are real values between 0 (poor or malicious) and 1 (good).

On completion of each transaction with the peer P_j, P_i evaluates the transaction and generates a number that reflects his satisfaction and appends it to the end of the Experience Vector associated with peer P_j. The vector stores the results of the most recent n experiences and so as new experiences are appended the oldest ones are removed. During the initialization phase all data items will be set to zero.

Voting

Resource Vote. The vote of peer P_i for a resource with ID ($resource_{id}$ is simply $\lambda_{resource_{id}}$). This allows the polling peer to learn precisely how the voting peer rated the resource.

Peer Vote. Voting for a peer uses the content of the Experience Vector associated with that peer. This information will be used to generate a vote that is a number in the interval $[0, 1]$. The function that is used to calculate the vote must cater for conflicting requirements. On one hand it must harshly treat peers who have resulted in a bad experience so that opportunities for malicious agents to share malicious resources while still enjoying a good reputation are reduced. On the other hand the system must provide tolerance for situations in which an otherwise good peer inadvertently shares a bad resource, perhaps by downloading it from a malicious peer and leaving it in its shared directory. In such a circumstance, the innocent peer should not be penalized to such an extent to exclude it from all subsequent transactions.

To reconcile the above conflicting requirements we will use a vote evaluation function that uses all elements of the $E(i, j)$ and reduces the effect of low ratings by using the square function. To generate the vote $\nu_{i,j}$ of peer P_i for peer P_j, we use the following:

$$\nu_{ij} = \sum_k \tilde{v}_{ij,k}/n$$

One may use other criteria, for example giving more importance to more recent experiences or emphasizing bad experiences. To implement the function in the former case higher weights (multipliers) may be used for more recent experiences, and for the latter case a higher power function (x^n), can be used.

Using the Experience Vector and the above method of calculating votes, provides a conservative method of admitting newcomers to the system with reputation built over time. Experience vectors are initialised to zero, resulting in votes for newcomers to be low until the experience vector for that peer is filled with real experience values. Using this method, a newcomer must make some effort in order to gain a good reputation. One way this can be achieved is for a newcomer to share popular resources that already have a good reputation. Similarly, new resources can build a good reputation by being offered by reputable peers. Malicious agents must also undertake this effort to gain a positive reputation and it is a primary consideration of the X^{2}Rep protocol that the speed with which malicious activity is identified, that this effort is not worthwhile.

Evaluating Ratings for Downloads. After a specified time period (set in a configuration file) the polling peer will have received zero or more PollReply messages (votes). The peer must now convert these votes into an evaluation for a possible transaction. Each transaction is specified by a $resource_{id}$ and a $peer_{id}$.

If rating evaluation is only based on the votes received from other peers, the polling peer implicitly assumes the voting peers are honest. However a received vote may have been spoofed, or may be generated as part of a malicious strategy used by a group of colluding peers. To reduce this implicit trust in voting peers we introduce an additional factor called *credibility*. Credibility focuses on the reliability of peers with respect to the voting process and is an indication of the confidence that a polling peer places in the votes provided by other peers. The experience vector reflects the satisfaction of P_i with respect to transactions with a specific peer and combines quality of the downloaded resource and the peer. The credibility c_{ij} is given by the peer P_i for the peer P_j that has provided votes in previous transactions and will be stored in the Local Credibility Repository of the peer P_i. Credibility c_{ij} is a real number in the interval $[0, 1]$ and is initialised to zero for an unknown peer.

On completion of a transaction, credibility of all the peers who had participated in the voting phase of the transaction are updated. The updating policy may vary. The aim of this policy is to reward the peers who have voted correctly, that is in accordance with the assessment of the transaction after the download, and punish those whose votes were contrary to this assessment. Voter credibility is updated for all peers who participated in the voting process, whether the peer voted for the resource, offering peer(s) or both.

Credibility values will be used to adjust a peers' votes for the current download. A peer P_i that sent a resource vote μ to polling peer P_j, will have the Adjusted Resource Vote $\tilde{\mu}_i$ as $\tilde{\mu} = \mu c_{ij}$. A peer P_i, that sent a peer vote ν_{ij} about offering peer P_j to peer P_l, will have the Adjusted Peer Vote $\tilde{\nu}_{ij}$ as: $\tilde{\nu}_{ij} = \nu_{ij} c_{li}$.

To collate these voting information to produce a single value for the rating of each entity (resource, or peer) one may use an average value. However this could result in attacks by large collusions to succeed. We noted that a collusion of malicious peers may degrade the reputation of an entity (resource or peer) by all providing bad votes during the polling phase. The X^{2}Rep protocol mechanism to detect and counter such activities is to identify agents that vote inaccurately and punish them by reducing their credibility to zero. As such, a vote cast by an agent with no credibility will have little effect on the summation the of adjusted votes. However, if we are not careful in this approach, a large number of votes of this nature will significantly reduce the calculated trust value. To negate this attack we choose not to include votes from peers for which there is no credibility rating or peers whose credibility rating is zero. Thus:

– Resource Trust Value $R\tau_i = \sum \tilde{\mu}_i$ where c_i is not 0
– Peer Trust Value $P\tau_j = \sum \tilde{\nu}_{ij}$ where c_i is not 0

$R\tau_i$ and the set of $P\tau_j$ will be used to find the most trustworthy offering peer and to determine whether the transaction can be considered trustworthy overall. A simple approach will be to use *threshold* values to define trust categories and determine the category of an entity with a given calculated trust by comparing it against this threshold. For example running averages of all previous $R\tau$ and $P\tau_{ij}$ can be maintained and the difference of the calculated trust and these values be used as to determine the category of the current entity.

The final trust value presented to the user will be a combination of the resource and peer trust values. The simplest approach would be to find the average of the two values. Users can use trust categories combined with other criteria, for example accepted level of risk, to make the final decision.

Updating State on a Peer. After the completion of a transaction, the state information of the downloading peer must be updated. This includes the following.

– Updating the downloading peer's Local Reputation Repository with peer and resource evaluation values.
– For each peer that provided a vote:
  - If the voting peer P_i provided an accurate vote $c_i = c_i + 0.05$.
  - If the voting peer P_i provided an inaccurate vote $c_i = 0$.

Although reducing c_i to zero for a single inaccurate vote may seem harsh but this will protect against a malicious agent who plans to build credibility and then subsequently use that credibility to provide inaccurate votes over an extended period. Reducing credibility to zero after a single inaccurate vote will minimise the success chance of such peers. This is an important feature as it is relatively easy to build credibility. Clearly, an unfortunate consequence is that a well-intentioned peer that mistakenly provides an inaccurate vote will have its credibility reduced to zero on the downloading peer.

5.2 XRep Protocol Modifications

X^2Rep requires that the PollReply message in *phase 2* of the XRep protocol be modified to include the identity of the sending peer. Thus we define the message as: PollReply ($\{[\text{peer}_{id}, \text{votes}]\text{Sk}_{voter,}\text{Pk}_{voter}\}\text{Pk}_{poll}$).

The voting peer sends it $peer_{id}$ and the set of votes signed using its private key Sk_{voter}, and its public key $\text{Pk}_{voter.}$The entire message is encrypted with the public key of the polling peer Pk_{poll} which had been sent in the Poll message.

This modification to the PollReply message is significant because it allows the polling peer to be certain that the $peer_{id}$ provided belongs to the voting peer and thus it can find the correct credibility rating. This assurance is made in what will now be *phase 3* of the XRep protocol as follows:

1. The polling peer decrypts the message using its private key $\text{Sk}_{poll.}$This provides confidentiality to the message and votes cast.
2. The polling peer verifies the signature of the $peer_{id}$ and votes token using the public key provided by the voting peer $\text{Pk}_{voter.}$This ensures that integrity of the token.
3. The polling peer finds the digest of the public key provided by the voting peer Pk_{voter}using a secure hash function and compares the result to the $peer_{id}$ provided by the voting peer. If these values are equal then the polling peer can be assured that $peer_{id}$ provided in the message really belongs to the voting peer.

The polling peer can then search its Local Credibility Repository for a rating for that voting peer. Obviously, there is nothing to stop a malicious agent creating a new $peer_{id}$ and key pair with which to vote, but the polling peer will have no credibility rating for that $peer_{id}$ and the vote will not carry any weight in the trust evaluation. We note that $peer_{id}$ is an opaque identifier and does not affect the anonymity provided by the underlying system.

As a result of this modification, a number of alterations can be made to the XRep protocol.

– Vote clustering techniques in *phase 3* of the XRep protocol are no longer required. Such techniques offer little protection against the activities of collusions of malicious agents. X^{2}Rep provides a more robust approach, combining the verification of the $peer_{id}$ provided in the PollReply message and the use of voter credibility ratings.
– The *TrueVote* and *TrueVoteReply* message exchange in *Phase 3* is no longer required as the verification of votes and authentication of the $peer_{id}$ are now provided in the PollReply message.
– For the same reason *phase 4*, the Best Peer Check message exchange is also not required.

These modifications significantly reduce the amount of message exchanges required by the protocol, whilst providing more a robust approach to combat malicious strategies. It should be noted that the X^{2}Rep algorithm requires an additional signature in the PollReply message. The extra computation required for this signature is not negligible but we argue that it still remains more efficient than the network communication that it replaces.

The XRep protocol with X^{2}Rep extensions is summarised in table 1.

5.3 Consideration of X^{2}Rep Properties

In this section we consider how the X^{2}Rep extensions measure against the properties of a reputation system, as defined in section 4.

Firstly we analyse the effects of the X^{2}Rep extensions on efficiency. A major design goal of X^{2}Rep is to reduce the amount of network communications required to determine trust. This is to ensure that the system is as scalable as possible within the constraints of the underlying architecture [9,3]. This is achieved by making redundant several unnecessary XRep network communication phases. This considerable benefit is reached as a result of a small amount of additional storage and computation resource use on each host machine. Given the processing power and amount of disk space available on modern PCs, we believe this trade off is reasonable.

Both XRep and X^{2}Rep fullfil the operational requirements of a reputation system. Neither system requires any modifications to the underlying network architecture or its dynamics. Furthermore, the inherent anonymity provided by the network is retained. This is achieved through the use of opaque identifiers, which cannot be traced to individual participants.

X^{2}Rep provides significant improvements over XRep in its security properties. X^{2}Rep utilises the cryptographic security features provided by XRep that prevent vote tampering. This is obtained through the use of digital signatures.

Table 1. XRep Protocol with X^{2}Rep extensions

Phase	Description
1 Resource Searching	*Poller to Network: Query (search_string, min_speed)* *Offerer to Poller: QueryHit (no_hits, IP, speed, ResultSet, trailer,* $peer_{id}$*)*
2 Resource Selection and Vote Polling	Poller to Network: Poll ((resource_id, $\{peer_1, \ldots .peer_n\}$, Pk_{poll}) *Voter to Poller: PollReply* ($\{[peer_{id}, votes]Sk_{voter}, Pk_{voter}\}Pk_{polr}$)
3 Vote Evaluation	Actions: – Find credibility ratings for voting peers and adjust votes accordingly. – Create Adjusted Resource and Peer votes and compare them with each other and with the Trust Threshold values.
4 Resource Download	Actions: – Select peer from which to download – After download, check the resource's digest – Update repositories and credibility ratings
Legend (Pk_i, Sk_i) pair of public and private keys where i can be a peer or poll request $\{M\}_k$encryption of message M under key k $[M]_k$ signature of message M under key k	

Furthermore, the smarter vote evaluation algorithm protects against generation of false reputations through the execution of malicious strategies. We have explored this improvement in our simulation. However, due to the page limitation, we omit the detail in this paper and we refer the reader to the full version of this paper.

X^{2}Rep continues to deliver the same safeguards as XRep against ID Stealth attacks. The primary modification is that X^{2}Rep requires that the offering peer provides both its public key and $peer_{id}$ within the PollReply message, and that the message is signed by its private key. Prior to initiating a download from a selected offering peer the downloading peer can carry out the same verification as previously described in the Best Peer Check phase of the XRep protocol.

6 Conclusion

We have presented X^{2}Rep, a trust semantics algorithm that extends the XRep reputation-based trust management protocol. X^{2}Rep provides reliable evaluation of trust even against a number of hostile environments. We started by describing the XRep protocol and identified some weaknesses. We then outlined our algorithm by describing the semantics for the storage of experience information, the generation of votes and the evaluation of trust. We gave detailed descriptions of how X^2Rep improves on the XRep protocol, including how two of the XRep message exchanges can be dropped. We then demonstrated through simulation that although the performance of a simplistic trust semantics algorithm is comparable to X^2Rep in an environment where no malicious strategies are imposed, X^2Rep displays far more accuracy and robustness when such strategies are introduced. Our conclusion is that when X^2Rep is integrated into the XRep protocol, a significant improvement in trust evaluation reliability can be obtained. Furthermore, this improvement is achieved whilst reducing network traffic and the complexity of the protocol.

References

1. Aberer, Cudré-Mauroux, Datta, Despotovic, Hauswirth, Punceva, Schmidt, and Wu. Advanced Peer-to-Peer networking: The P-grid system and its applications. *EPFL Technical Report IC/2002/73*, 2002.
2. Aberer and Despotovic. Managing trust in a Peer-2-Peer information system. *Proc of the Ninth International Conf on Information and Knowledge Management*, 2001.
3. Anderson. Analysis of the traffic on the gnutella network. March 2001.
4. Cornelli, D. C. di Vimercati, Paraboschi, and Samarati. Choosing reputable servents in a P2P network. May 2002.
5. Damiani, C. di Vimercati, Paraboschi, Samarati, and Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. November 2002.
6. Kamvar, Schlosser, and Garcia-Molina. Eigenrep: Reputation management in p2p networks. 2002.
7. Resnick, Zeckhauser, Friedman, and Kuwabara. Reputation systems: Faciliating trust in internet interactions. 2000.
8. C. D. S. Services. The gnutella protocol specification v0.4 document revision 1.2.
9. K. Sripanidkulchai. The popularity of gnutella queries and its implications on scalability. 2001.