Adaptive Content Poisoning To Prevent Illegal File Distribution in P2P Networks[*]

Xiaosong Lou and **Kai Hwang,** *Fellow IEEE*
University of Southern California, {xlou, kaihwang}@usc.edu

Abstract: Digital content owners have attempted to use content poisoning to disrupt illegal distribution of copyrighted files in *peer-to-peer* (P2P) systems. This paper provides an analytical model to quantify the impact of content poisoning. Tradeoffs between content poisoning and download efficiency are revealed. In particular, we apply this poisoning model to transform and evaluate three popular P2P networks: *BitTorrent, eDonkey, and eMule,* for copyright-protected content delivery. We find that chunking protocol, hashing scheme, peer selection, decoy number, and decoy blacklisting play crucial roles to prevent illegal file distribution over P2P networks.

We discover that BitTorrent is most resistant to content poisoning. Index poisoning could be a viable alternative to cope with copyright violation on BitTorrent. The eDonkey is less resistant to content poisoning than eMule. A new *challenge-response protocol* (CRP) is proposed using majority-vote on multiple chunk copies received. To control the level of poisoning, we adapt *repeated peer selection* (RPS) and *decoy blacklisting* techniques. Both eDonkey and eMule can apply CRP, RPS, and blacklisting to perform fine tuning of the content poisoning process. These mechanisms can transform eMule to become less resistant than eDonkey, which will serve the best interest of content owners. Finally, we elaborate on integrated research tasks needed to achieve seamless copyright protection in P2P networks.

Keywords: ***Peer-to-Peer systems, content delivery, copyright protection, content poisoning, chunking protocols, decoy deployment, distributed file sharing, P2P reputation systems, download efficiency,*** and ***digital right management.***

---

[*] Manuscript submitted to *IEEE Transactions on Computers*, August 21, 2006. This work was supported by NSF Grant ITR-0325409 at the University of Southern California. All rights reserved by the authors and publisher. Corresponding author is Kai Hwang at Email: kaihwang@usc.edu, Tel. 213 740 4470 and Fax: 213 740 4418.

1. Introduction

Peer-to-peer (P2P) file-sharing systems are the most popular Internet content-delivery systems in use today [8] It was reported that P2P content delivery constitutes a large portion of the Internet traffic[35] .Currently, popular P2P content delivery systems include the eDonkey/Overnet, BitTorrent, eMule, Gnutella, KaZaA and others [20] . While these systems are widely used for distributing open-source contents such as Linux operating systems, a significant portion of the contents being delivered are music and movies that are possible copyright violations.

Illegal distribution of copyrighted material over the Internet has been a serious problem long before the existence of Napster and KaZaA. However, the anonymity nature of P2P network makes them most attractive to Internet pirates. Although the P2P file-sharing technology was not invented to facilitate illegal distribution of copyrighted material [31] , it has become a primary means for on-line piracy. Copyright protection issues in P2P file-sharing systems include *decoy blacklisting, smart peer selection, digital rights management* (DRM), *content poisoning,* and the use of *reputation systems,* etc.

In this paper, we consider only the situations that clients get the proper fileID using the www or email services. The deliberate falsification of file content is referred as *poisoning*, while accidental corruption of file content is referred as *pollution* [21] . This definition coincides with those defined by Christin el al [5] . A similar technique, *index poisoning* [22] , refers to the strategy that decoy providers make false responses during the query stage and clients are pointed to the wrong host or wrong file. Index poisoning is beyond the scope of this paper.

To apply content poisoning, the copyright owner sets up decoy peers that respond to illegal download requests with corrupted or falsified files. The rationale behind such a deterring technique is that if the unpaid clients keep downloading falsified file, eventually frustration will lead them to stop the abusive use of P2P file-sharing services. Content poisoning is attractive because it only targets individual file in the P2P network. It does not affect legitimate use of P2P file sharing systems. While poisoning technology certainly causes some increase of network traffic, the drawback only occurs on those peers that are actively sharing the targeted file.

Content poisoning is also known *denial-of-service resilience* [9] . This paper attempts to point out the cause behind the disparities in effectiveness of content poisoning via quantitative analysis and simulation experiments. In order to measure the impact of content poisoning in different P2P file sharing systems, a new performance metric called *poisoning effect* is introduced. Using this metric, different poisoning resistance level of different P2P file-sharing systems are identified and compared. We evaluate and anticipate that P2P systems will take steps to improve their resistance to content poisoning by protocol improvement, repeated peer selection, and decoy blacklisting, etc. We make recommendations each chunking protocol under various network conditions.

The rest of this paper is organized as follows: Section 2 reviews recent reports of related work. Section 3 introduces a new metric to quantitatively measure content poisoning effects. We also characterize three popular P2P File Sharing Systems in Section 3. Section 4 introduces a new protocol for supporting adaptive content poisoning in chunked file delivery. In Section 5 we model the poisoning effects of various P2P systems. Section 6 reports experimental results in evaluation poisoning effects. Based on these results, we present our guidelines for adaptive content poisoning in section 7. Finally, we summarize the contributions and suggest further research needed.

2. P2P Content Delivery and Related Work

As one of the earliest protocols, the FTP still remains popular and successful 30 years after its inception. Later on, the HTTP protocol dominated Internet traffic in more recent years [35] . By late 1990's, *content delivery networks* (CDN) [18] became popular as the second generation of Internet Content Delivery. The third generation is the P2P file-sharing systems [15], [35]. Unlike the earlier two generations, the P2P file-sharing systems are mostly home grown; not supported by existing Internet protocols.

2.1   Evolution of Digital Content Delivery

Saroiu et al [31] has classified Internet content delivery systems into three generations. The first generation is for *client-server architecture* using FTP and HTTP. The server delivers contents upon request from client. Earlier implementations require the download process be completed within a single session. Later a download can be paused and resumed by the client. The client could also choose to open multiple sessions to the same server and download multiple files in parallel. We summarize in Table 1 the architectures, applications, and representative systems of three generations of content delivery systems.

Table 1 Three Generations of Digital Content Delivery Systems

System Generations	Network Architecture	Download Bandwidth	Content applications	Example Systems and web sites
First Gen.: Client-Server system [32]	Clients access server using FTP and http	Limited by single server bandwidth	Small files with limited download request	Apache http://apache.com/ Getright http://getright.com/ CuteFtp http://cuteftp.com/
Second Gen.: Content Delivery Networks [15,18]	Clients connect to multiple servers	Load-balanced among globally located servers	Time-sensitive video streaming, virus signaturex	Akamai: http://akamai.com/ SyncCast: http://synccast.com/
Third Gen.: P2P File-Sharing System [, 35]	Peer-to-peer overlay network	Low, peers are not secured and less reliable	Large files, download speed is not critical	BitTorrent http://bittorrent.org/ eDonkey http://edonkey.com/ eMule http://emule.org/

The second generation of Internet content delivery is called *content delivery networks* (CDN) [15] [18] [28] . They use the same client/server protocol as the first generation, but their network consists of large number of globally located servers [15]. These servers replicate contents via caching and deliver content to clients according to geographical vicinity and bandwidth availability. In [29] , Pierre et al. proposed an optimal technique that allows each file to have its own replication strategy. Mei, et al [25] also studied this problem.

CDN improves content availability because a disrupted server will not stop the entire delivery service. It also improves response time and throughput [18] because the client will dynamically connect to a server that has lowest workload and/or network latency. CDN is still the main architecture in delivering critical, time sensitive contents such as operating system patches and virus signature files [15] .

The third generation of content delivery systems is the *P2P file-sharing system*s. These systems do not require a server to deliver contents. In some cases (BitTorrent [30]) an index server is present to answer queries and direct traffic to peers. In other cases such as eDonkey/Overnet, the entire network is built on *Distributed Hash Table* (DHT) overlays, that even index server is no longer required. This generation is clearly superior to previous generations in terms of content availability. Because content is distributed from peer to peer, it only incurs the minimum cost to content distributor.

However, P2P users often experience longer response time and slower download, because QoS is not guaranteed in P2P systems. P2P networks are suitable to distribute ultra-large files (hundreds of Megabytes or more). Smaller files are often distributed in traditional FTP or HTTP servers due to higher throughput and lower cost at the server. It is ineffective to distribute unpopular files in P2P systems because only popular files will attract enough peers.

2.2 Related Work on Copyright Protection in P2P Systems

P2P file-sharing systems are mostly built for music and video content delivery with huge traffic volume across the Internet. P2P content poisoning has been a controversial topic for copyright protection [5, 21, 32], because content owners and client users have conflicting interest. We review the previous related work in this section. Content poisoning has been patented [14] and implemented to fight against Internet piracy. However, several poisoning attempts by content industry were not successful [26] , [27] . The demand is mount to have a good theoretical model to assess poisoning effectiveness in P2P Networks [4, 15, 21, 28, 31, 33]. This paper is meant to provide such a model to amend the problem.

Mei, et al [25] and Zanin et al [41] proposed an adaptive algorithm for distributed file fragment locating in large-scale distributed file systems. Yurkewych et al [40] presented a game-theoretic model to investigate the cost-ineffectiveness and redundancy in commercial P2P computing. Manini et al [24] modeled the performance of P2P file sharing applications. They indicated that the smart selection of fast and reliable peers is a key to improving the downloading performance. Bernstein et al [2] and Wu and Li [38] have suggested different peer-selection methods. Their ideas are based on advertised resource availability at peer sites. Gedic and Liu [12] proposed the PeerCQ, a P2P architecture for Internet information monitoring applications. Xiao et al [39] proposed an algorithm that solves the topology mismatching problem caused by peers randomly joining and leaving the network.

If a P2P file-sharing system is suspected of poisoning, the advertised resource availabilities of a peer are no longer credible. This problem has led to the introduction of reputation systems into P2P file sharing [7] . Kamvar et al [16] introduces the EigenTrust reputation system. Damiani et al [6] proposed a distributed polling algorithm so that clients can have access to peer reliability information before download starts. Gupta et al [13] designed a public key based mechanism that periodically updates peer's reputations in a distributed manner. Walsh and Sirer [37] proposed an object-based reputation system, Credence, which can counter SPAM and poisoning. Zhou and Hwang [42] introduced the PowerTrust system supporting trusted P2P applications.

DRM systems [11] [23] were developed to distribute digital contents to authenticated clients. Most of the DRM systems are based on cryptography [4] . Unfortunately, to ensure portability and fair-use to all users, digital contents eventually have to be consumed in plaintext. Perpetrators can record the plaintext and convert DRM protected contents into pirated copies. To amend the limitation of using expensive DRM system, content poisoning was suggested for protection of copyrighted files in P2P systems. This technique has been used by publishing industry to fight against illegal distribution of digital contents over the Internet.

Christin et al [5] conducted a series of experiments on popular P2P file-sharing systems including eDonkey, eDonkey/Overnet, FastTrack and Gnutella. Their findings indicate a strong correlation between content availability, network topology, and a small number of decoys can have significant impact on peers' perception of content availability in a P2P system. Dumitriu et al [9] evaluated different P2P network topologies and decoy or client client strategies. They discovered that randomized peer selection strategy effectively resist poisoning. However, thers is a need to explore deeper into the file chunking protocols, a quasi-standard behavior among real P2P file delivery systems. To amend this shortcoming, our work focuses on the effects of different file chunking protocols for adaptive content poisoning.

3. Chunked Content Poisoning Protocols

As the shared file sizes tend to become large, most P2P systems adopt file-chunking methods. For users, file chunking enables parallel download, which increases download throughput. It also allows peers to share part of a file before the peer receives the entire file, which improves file availability among peers. On the other hand, file chunking improves the effectiveness of content poisoning because the owner needs to poison only a single chunk to corrupt the entire file.

3.1 P2P Content Poisoning Environment

Currently, content owners only distribute free-of-charge files such as product demo, freeware and open-source software via P2P file-sharing networks. They do not distribute content to paid customers via P2P networks due to the high risk of copyright infringement. If any of the paid content is found in P2P networks, they are pirated. In that case, content owners will use content poisoning to disrupt the illegal file distribution. The disruptions appear as failed distribution, corrupted contents, or repeated frustrations on the part of unauthorized downloading clients. The concept of content poisoning in such a P2P system is illustrated in Fig.1.

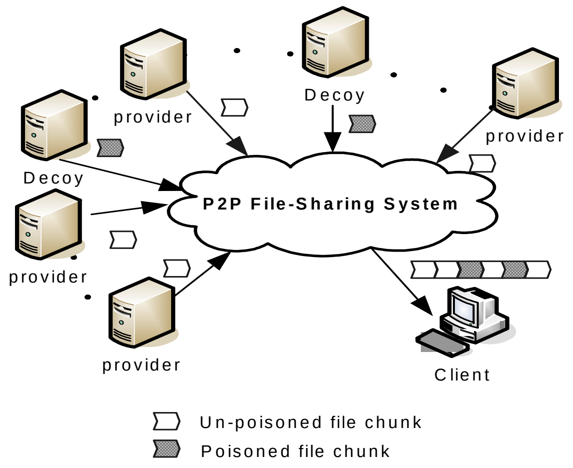


Figure 1: Content poisoning in a P2P file-sharing System, where providers generate clean file chunks and the decoys generate poisoned chunks

The illegal content *providers* on a P2P network are the copyright violators. They share clean file chunks to other peers. The content owners deploy a small number of *decoys* to poison the copyrighted files at subdivided chunk level. Unlike providers, decoys send out poisoned chunks in response to client's download requests. In the sequel, we use the term *peer* to refer to either a provider or a decoy, because both are not distinguishable by clients. The number of decoys divided by the total peer count is the *decoy density*.

For a real example in August 2006, we discovered a piracy case, by which a brand-name word-processing package was illegally distributed over a well-publicized P2P network. We hide the real names and fileID involved in order to protect all the parties involved. We refer the software file by a fake name: *XYZ Pro*. This file was measured to have 643.63 MB, subdivided into 3571 chunks. The file is identified by a fake fileID =*XYZ1234567890Pro*. When requesting the file, 56 peers on the P2P network responded, among them 8 peers are immediately available for file transfer. Based on observing many similar cases, we set up the simulation parameters in Section 6.

By deploying decoys into P2P file-sharing systems, owners want to discourage the download of copyrighted files by all clients, who did not subscribe the requested file. The major design issue of content poisoning lies in cost-effective deployment of decoys. Effective poisoning should use only a few decoys. Excessive decoying is unnecessary and cost-prohibitive. Content owner should also anticipate that those clients download pirated content off the P2P network will apply various techniques to resist poisoning.

3.2 Definition of Content Poisoning Effect

Many technical factors affect the download performance of a P2P file-sharing system, such as network topology, routing scheme, traffic congestions over the network, geographical locations of peers, etc. In order to single out and focus on the effect directly caused by content poisoning, we need to filter out all other effects. Let S be the actual file size (in Mega bytes) and D be the total number of bytes downloaded. We define the *poisoning effect* by:

$$\textit{Poisoning Effect} = 1 - S / D \qquad \textbf{(1)}$$

Poisoning Effect isolates the download effort wasted due the existence of decoys providing poisoned chunks. Its value represents the portion of downloaded bytes that are wasted due to the existence of decoys in a P2P file-sharing system. For example, in an ideal P2P file-sharing system where no decoy was present, we have $S = D$, meaning the client received exactly the same amount of bytes as the actual size of the file. Thus, the poisoning effect becomes zero. The poisoning effect approaches 100%, if D becomes extremely large, meaning most download requests failed.

The content owner does not care of the copyright violator's interest. Their goal is to achieve higher poisoning effect. For example, 0.9 poisoning effect implies that the client must download 10 copies of the same file in order to retrieve an un-poisoned version. Since the owner's cost is directly related to the number of decoys deployed, a cost effective approach is to deploy just enough decoys.

3.3 File Chunking Protocols

Resent survey conducted by Cachelogic company [3] shows that the three most popular P2P file-sharing systems are BitTorrent [17] , eDonkey [10] and eMule [19] . In these P2P networks, a large file is sliced into predetermined chunks. These file chunks are hashed to generate the index file (BitTorrent) or fileID (eDonkey and eMule). For P2P file-sharing clients, file chunking enables parallel download, which increases download throughput. It also allows peers to share part of a file before the peer has the entire file, which improves file availability among peers. From content owner's point of view, file chunking improves content poisoning because the content owner only needs to poison one file chunk to corrupt the entire pirated file. We compare these three file-chunking protocols Table 2.

Table 2 Three File Chunking Protocols used in P2P Content Delivery

Chunking Protocol	Chunking Scheme	Hash Algorithm	Hash Distribution	Poisoning Detection
BitTorrent [17] , [30]	Divide files into (64KB ~2MB) pieces and then subdivide into (8 KB ~ 32 KB) chunks	SHA	Hash tree in index file and distributed outside of P2P network	Detectable at chunk level
eDonkey [10]	Subdivide files into parts (9500 KB) and further divided into 180KB chunks	MD4-based	Concatenated chunk hashes form the fileID	Detectable after download the entire file
eMule [19]	SubdivIDe files into parts (9500 KB) and further divided into 180KB chunks	MD4-based	FileID generated from chunk hashes; peers exchange part hashset	Detectable only if part hashset is not poisoned

A. The BitTorrent Protocol

In BitTorrent protocol, each file is divided into pieces, and each piece is further divided into chunks. Typical piece size is between 64 KB and 2 MB. Typical chunk size is between 8 to 32KB. The chunking parameters and hash values of all the pieces are stored in an index file (also called the metadata). The client needs to acquire the index file from FTP/HTTP or email, before starting download.

Because the index file is distributed outside of the P2P file-sharing system, each chunk can be verified with a reliable hash contained in the metadata. This verification provides BitTorrent protocol with high resistance to content poisoning. This static chunking scheme does present another problem. As the files become larger, so does the size of the metadata. In many cases the the metadata could exceed 100KB. Increasing the piece size would provide a temporary solution, but as piece size grows too large, the verification of each piece becomes slower.

B. The eDonkey Protocol

Similarly to BitTorrent, eDonkey divides each file into 9500KB parts, and each part is further subdivided into 180 KB chunks. The hash value of these chunks are concatenated into a long string and again hashed into a global hash contained in fileID as detailed in [10] . Unlike BitTorrent, however, eDonkey clients are not able to verify the chunks until the entire file have been received, because the client only has the hash of all hashes and the one-way hash function implies that hash value of any single chunk cannot be derived from the fileID.

Like eDonkey, many other P2P file-sharing protocols do not have a reliable hash tree at the client side. Content poisoning is very effective in these protocols, because poisoned chunks cannot be detected until the entire file is received. Furthermore, the client that unknowingly receives a poisoned chunk will propagate that chunk, and further poison other peers. Finally, when the client detects corruption of file via hash mismatch, it cannot identify the chunk that was corrupted; thus the entire file must be discarded and downloaded again and again.

C. The eMule Protocol

Because of the popularity of eDonkey, many modifications have been introduced. The eMule [19] is the most popular upgrade from eDonkey. This protocol allows peers to exchange the *part hashset* (containing hash values of all the parts), so that the client can verify a part when it is received. Because the part hash values are distributed, they can be poisoned just like any regular file chunk. Current implementation of eMule only allows exchange of hash value on part level, or every 53 chunks. When the client detects a corrupted part, it only downloads one chunk at a time within the part and re-computes part hash, until the hashes match.

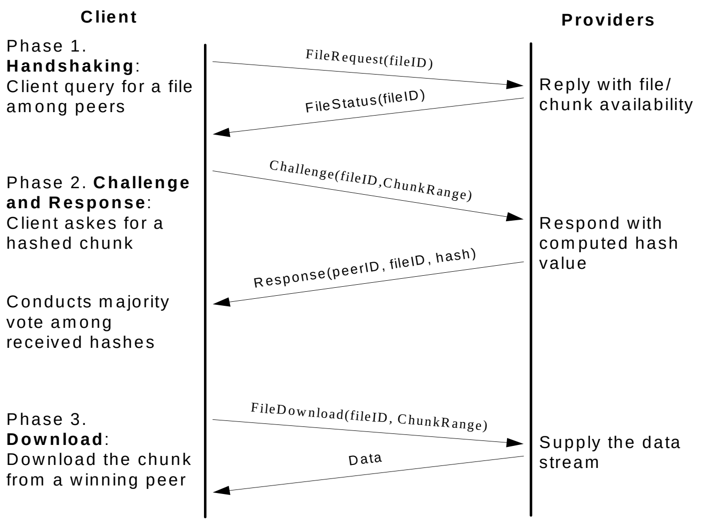
Because the exchanged hashset contains hash values for all parts and the fileID contains global hash, verification of the part hashset against the fileID is not possible. Furthermore, the exchange of part hashset relies on the replying peer having the full part hashset. If the peers compute its own part hashset, no part hashset can be exchanged until a peer has all the parts of the file. On the other hand, if a peer is allowed to pass along the part hashset it receives from other peers, the decoy can deliberately propagate a poisoned part hashset to disrupt the entire eMule network.

4. CRP Protocol, Peer Section, and Decoy Blacklisting

This section introduces a new protocol for adaptive content poisoning. This protocol is applicable to transform or to relax the eDonkey and eMule in their resistance to poisoning effects. Because P2P users can select preferred peers and use blacklist to identify decoys, we introduce how to cope with these resistant situations.

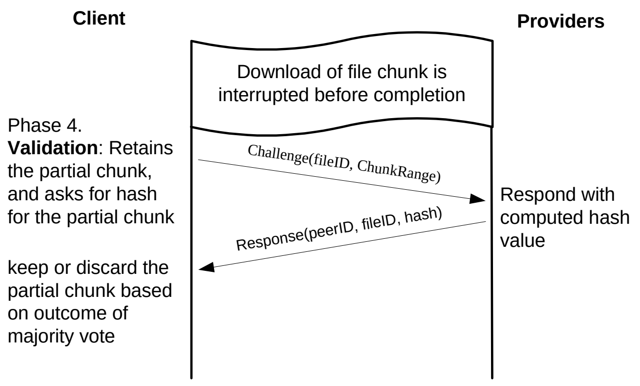
4.1 The CRP Protocol

We propose a dynamic file chunking scheme supported by a *challenge response protocol* (CRP). The protocol allows a partial file of any size be treated as a chunk and joined with the rest of file. In order to verify file authenticity (against poisoning) and integrity (against pollution), the client exchanges a set of challenge/response messages with peers during download. Adaptive file chunking does not require fixed chunk sizes. Through majority voting, it gave the client a better chance to select clean providers over poisoning decoys. Because the chunk size and location is determined at run time, decoys that answer with random hashes will easily be voted off. Figure 2 shows the 4 phases of the CRP protocol.



Client
Providers
Phase 1. **Handshaking**: Client query for a file among peers
FileRequest(fileID)
FileStatus(fileID)
Reply with file/ chunk availability
Phase 2. **Challenge and Response**: Client askes for a hashed chunk
Challenge(fileID,ChunkRange)
Respond with computed hash value
Response(peerID, fileID, hash)
Conducts majority vote among received hashes
Phase 3. **Download**: Download the chunk from a winning peer
FileDownload(fileID, ChunkRange)
Supply the data stream
Data

(a) Download without interruption



Client
Providers
Download of file chunk is interrupted before completion
Phase 4. **Validation**: Retains the partial chunk, and asks for hash for the partial chunk
Challenge(fileID, ChunkRange)
Respond with computed hash value
Response(peerID, fileID, hash)
keep or discard the partial chunk based on outcome of majority vote

(b) After download is interrupted

Figure 2 Phases 1-3 of CRP specify uninterrupted client-provider handshaking. Phase 4 specifies a procedure to retrieve the winner chunk of the majority-vote outcome

Phase 1: Handshaking: The client inquires about the availabilities of the file in interest to all peers, and decides the size and location of the file chunk according to replies.

Phase 2*: *Challenge and response: The client sends a select group of peers a message requesting the hash of the dynamically decided file chunk (the Challenge). Selected peers reply with requested hash (the Response). Upon receiving the Response messages, the client conducts a majority vote. Those peers that win the vote are considered to have the authenticated copy of the chunk; others are considered to be poisoned.

Phase 3: Download: The client chooses the fastest peer among the majority for the chunk.

Phase 4: Validation: If the downloading of the chunk was disrupted for any reason, the client does not discard the partial chunk. Rather, the protocol computes the hash on the received part and sends a challenge message to all peers for the partial chunk. A majority vote is conducted among all chunks received and compared with the local hash. If they match, the chunk joins the rest of the file. Otherwise the questioned chunk is discarded.

Through majority vote, the CRP gives the client better chance to select provider rather than decoy. On the other hand, the content owner can increase the number of decoys beyond 50% density so that decoys have better chance to win the vote. Because the chunk size and provider location is determined at run time, decoys that answer with random hashes will easily be voted off in using the CRP. Furthermore, if the file chunk does not verify with the voted chunk hash, a decoy is immediately detected. CRP will favor owners when the decoy number increases to become the majority after reaching an agreement on a common poison.

Another advantage of this protocol comes from the fact that the client can retain partial chunks when a download is disrupted. This may not be significant in broadband connected peers. But as peers became scarce and less likely to stay online, every part of the file saved lead to quicker download. This protocol pays the cost of extra computation of dynamic chunk hashes. However, because the computation occurs at providers' side, its overhead is distributed among those providers. The extra traffic volume caused by CRP is also ignorable because hash size is very small compared to the file downloaded.

Owners inject decoys into the P2P file sharing networks. These decoys send the poisoned file chunks to the client. In a P2P file sharing system that does not use CRP to verify hash values, the decoy will simple send poisoned file chunk whenever it is requested. In a P2P network adopts chunk verification via hash values, a decoy will have only two options during the challenge/response phase. It can either send the hash computed from the un-poisoned file (clean hash) or from the poisoned file (dirty hash) as a response.

When a decoy responds with clean hash, wins the majority vote, and subsequently chosen as the provider, the client will receive poisoned file chunk. It can detect poisoning as soon as the chunk is received due to hash mismatch. In case the decoy responds with the dirty hash and wins the majority vote, the client cannot detect the poisoning until the entire file is downloaded. The advantage CRP lies in adaptive adjustment in the majority vote. As expected, the result of CRP relates tightly to both providers and decoys responded to download request at run time.

4.2 Repeated Peer Selection and Decoy Blacklisting

There exist other techniques to increase poison resistance in P2P networks. One such technique that is called *repeated peer selection* (RPS). Because of the worry of connecting to unreliable peers or poisoning decoys, rather than randomly chosen peer, the client software will pick the current provider for further file transfer, if those file chunks are available at that peer. This technique reduces the peer selection overhead during downloading the entire file. Also, the selection scheme may result in decoys less likely to be chosen by the client.

Another important technique is called *decoy blacklisting.* Traditionally, a blacklist contains blocked IP addresses including malicious peers, SPAM mails, and virus spreaders. The client identifies these IP addresses manually. This list is generally unreliable, because it changes with respect to time and network conditions. Dynamically maintaining such a blacklist is also very difficult. For P2P networks, the user may not be aware of all decoys, because they are supposed to be hidden and dynamically changing. In some cases a common provider may be erroneously blacklisted. These two techniques are available in some P2P file-sharing software. In section 6, we will show experimental results on the effects of RPS and blacklisting on content poisoning.

5. Modeling of Poisoning Effects

We model the poisoning effects of different file-chunking protocols in this section. The underlying file chunking scheme protocols determines the protocol’s resistance to content poisoning. Using poisoning effect, we can focus on the interplay between file protection and download efficiency, while ignoring other factors such as network topology, traffic congestions, peer locations, etc. To normalize our discussions, we consider a *chunk* the smallest part of a file, whose hash value is used to verify its authenticity and integrity. For example, the term of chunk corresponds to *piece* in BitTorrent and to *chunk* in eDonkey and eMule. Table 3 lists important parameters used in our analytical modeling of poisoning effects.

Table 3 Parameters and Notations used in Paper

Notation	Brief Definition
n	Number of content providers (copyright violators)
d	Number of decoys deployed by content owner
S	Actual file size in mega bytes (MB)
D	Number of bytes actually downloaded
c	Chunk size in bytes
m	Number of chunks in a file ($S = mc$)
θ	Poisoning effect defined by 1-S/D

5.1 Poisoning Models of File Chunking Protocols

Mathematically, the *decoy density* is defined by the ratio $d/(n+d)$. In this section, we model the poisoning effect of BitTorrent, eDonkey, and eMule protocols, separately. In BitTorrent downloading an entire file can be treated as downloading m independent chunks of size c each. If the peer is a provider, the total number of bytes downloaded for the chunk will be c. If the peer is a decoy, then after downloading c bytes, the client will detect the poison and restart the download process again. This leads to the following Theorem 1.

Theorem 1:

For the BitTorrent protocol, the poisoning effect equals the decoy density.

$$\theta = d/(n+d) \qquad \textbf{(1)}$$

Proof: The probability of selecting a decoy is $d/(n+d)$. For every chunk, we compute the expected number of bytes to be downloaded as $E_c = c[1- d/(n+d)] + (c+E_c)d/(n+d)$. Solving this equation, we have: $E_c = c(n+d)/d$. Because all m chunks are independent, the total number of bytes need to be downloaded is $D = mE_c = S(n+d)/d$, where $S = mc$. Therefore, the poisoning effect $\theta = 1- S/D = d/(n+d)$ is proven. ***Q.E.D.***

The eDonkey protocol covers all other chunking protocols that use no hash tree. A client cannot verify any part of the file until the whole file is received. If the file is poisoned, the client has to discard all the bytes received and start all over again. This leads to the following result.

Theorem 2:

For eDonkey protocol, the poisoning effect θ equals the probability that more that one file chunk(s) are poisoned.

$$\theta = 1 - [n/(n+d)]^m \qquad \textbf{(2)}$$

Proof: If none of the chunks are poisoned, then total number of bytes downloaded will be *S*. If any chunk is poisoned, the client cannot detect the poison until all *S* bytes are received. And all *S* bytes will have to be discarded. If a file contains *m* chunks, the probability that no of the chunks are poisoned is $[n/(n+d)]^m$. We compute the expected number of bytes to be downloaded $D=S[n/(n+d)]^m+(S+D)\times\{1-[n/(n+d)]^m\}$. Solving this equation, we have: $D=S/[n/(n+d)]^m$. Therefore, $\theta=1 - S/D = 1 - [n/(n+d)]^m$ is proven. ***Q.E.D.***

In eMule Protocol, the peers exchange part-level hashset. The client can verify 53 consecutive chunks against the hashset. The hashset is poisoned with a probability $d/(n+d)$ and $\theta = 1$. The hashset is not poisoned with a probability 1- $d/(n+d) = n/(n+d)$ and its θ is equivalent to that of the eDonkey protocol in Eqn.(2). Averaging the two cases, we obtain the following result.

Theorem 3:

For eMule protocol with part-evel hashset exchange, the poisoning effect is equivalent to that of eDonkey protocol when downloading a 54-chunk file.

$$\theta = 1 - [n/(n+d)]^{54} \qquad \textbf{(3)}$$

Proof: Files in eMule are usually much larger than a single part (9500kB). Due to the part-level hashset exchange mechanism, we analyze each part independently. For a part or 53- chunk file, the overall poisoning effect is the weighted sum of the above two cases. Therefore we have poisoning effect $\theta = d/(n+d) + n/(n+d) \times\{1- [n/(n+d)]^{53}\} = 1 - [n/(n+d)]^{54}$

Q.E.D

The above models suggest that decoy density, $d/(n+d)$, and the number of chunks (m) in a requesting file are two primary factors in deciding poison effectiveness. The poisoning effect of BitTorrent grows linearly with decoy density. The eDonkey increases exponentially as chunk number increases. The eMule offers a higher level of poisoning resistance than eDonkey. In effect, the eMule protocol puts an upper bound on poisoning effect in eDonkey. This bound is equivalent to download a 54-chunk file in eDonkey as seen in Equation (3). Base on these models, we conclude that content poisoning is more effective in eDonkey network than in eMule.

5.2 Poisoning Model of The CRP

The CRP protocol uses a simple majority vote scheme to improve the chance of a client selecting a peer, not a decoy, to download the requested file chunk. Intuitively, the CRP decreases the poisoning effect, when decoys are in the minority group. In a file chunking protocol, the optimal strategy for a decoy is to respond with false hash and win the majority vote. All decoys must agree on a common false hash to increase their chance of winning the vote.

Theorem 4:

In a file chunking scheme using the CRP protocol, the poisoning effect is related to the outcome of majority vote, which is computed by:

$$\theta = 1 - \left[\sum_{i=0}^{n/2} \binom{n}{i} \left(\frac{d}{n+d}\right)^{i} \left(\frac{n}{n+d}\right)^{(n-i)}\right]^{m} \qquad \textbf{(5)}$$

Proof: If n providers respond to each challenge, then the probability q of a true hash winning the majority vote is decided by a binomial distribution: $q = \sum_{i=0}^{n/2} \binom{n}{i} (\frac{d}{n+d})^{i} (\frac{n}{n+d})^{(n-i)}$. We calculate the conditional expectation of downloaded bytes by $D = Sq^m + (S+D)(1-q^m)$. Solving this equation, we have: $D=S/q^m$. Therefore, $\theta = 1 - S/D = 1 - q^m$ ***Q.E.D.***

Theorems 1 to 4 are obtained under two fundamental assumptions: (1) The decoys are hidden. They cannot be distinguished from the piracy providers. (2) The peers are randomly selected for content delivery services. In the next two sections, we will verify these analytical results by simulation experiments. We will also report experimental results that are relaxed from these assumptions.

6. Experimental Results on Poisoning Effects

In this section, we evaluate the poisoning effect of three most popular P2P file-sharing protocols via simulation experiments. The simulator was written in Perl script and run on a 4-node server cluster at USC Internet and Grid Research Laboratory. All the parameter settings in P2P network experiment are summarized in Table 4. We set simulation parameters very similar to the real-life example given in Section 3.1.

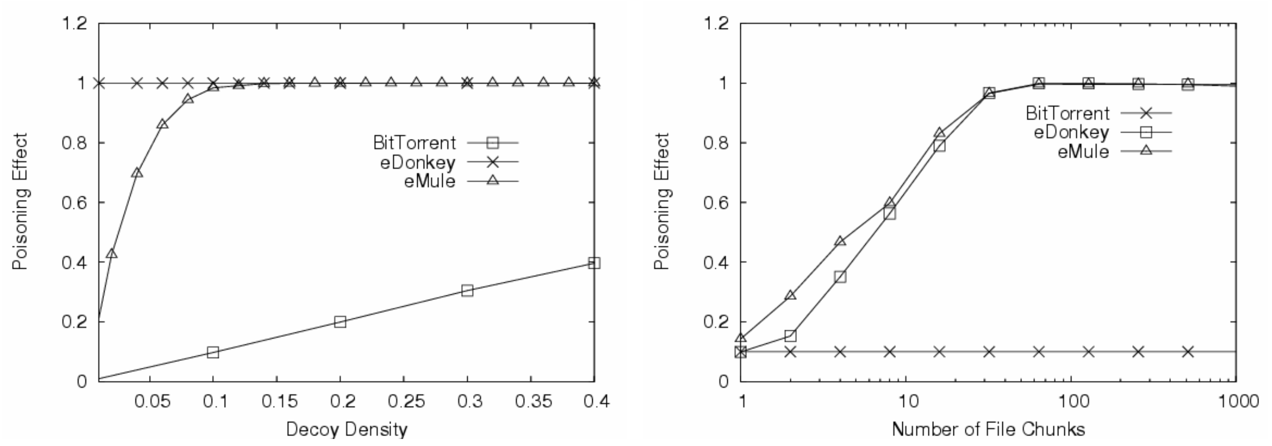
Table 4 Parameter Settings in P2P Network Simulation Experiments

Parameter	**BitTorrent**	**eDonkey**	**eMule**
Actual file size (***S***)	64MB ~ 2GB	180MB	180MB
File Chunk Size (***c***)	64KB ~ 2MB	180KB	180KB
Number of file chunks (***m***)	1,000	1,000	1,000
Total number of peers (***n+d***)	100	100	100
Number of providers (***n***)	0 ~ 100	0 ~ 100	0 ~100
Number of decoys (***d***)	1~ 99	1 ~ 99	1~99

Based on the settings in Table 4, there are at most 100 providers of a given file. We investigate up to 1,000 chunks in each file. A 1,000-chunk file is equivalent to 64 MB ~2 GB in BitTorrent or 180 MB in using eDonkey and eMule systems. Because poisoning effect is closely related to chunk number, not the file size, such a setting better presents differences between chunking protocols. The simulator implements different file-chunking protocols augmented with CRP, RPS, and decoy blacking options. It generates providers and decoys according to simulation settings. We request to download each file 100 times and report the average poisoning effects after 100 runs and measurements.

6.1 Effects of Decoy Density and Chunk Number

Both number of decoys used and the subdivided chunk number of a requested file affect the poisoning effect. We report the simulation results of these effects for three P2P networks. The simulation results on poisoning curves plotted in Fig.3(a) coincide with those plotted using the theoretical results specified by Eqns. (1–3). This verifies the correctness of the modeling formulations.



a) Effects of decoy density in P2P network **(b) Effects of chunk number in a file**

Figure 3 Poisoning effects of BitTorrent, eDonkey and eMule protocols in a simulated P2P network with a total of 100 providers and decoys per each file request.

The BitTorrent protocol is most poison resistant. This is caused by the fact that all file chunks are hash-verifiable against the index file. Because the index file is distributed to client out-of-band, it is not subject to content poisoning. Its poisoning effect increases slowly and linearly from 0 to 40%, as the decoy density increases to 40%. The eDonkey has the perfect poisoning effect with θ close to 1, because no file chunk can be verified. The poisoning effect of eMule increases from 0.2 to 1 after 10% decoys are used. Compared with eDonkey, eMule appears to be more resistant to content poisoning.

Figure 3(b) plots the poisoning effects against the chunk number in using the BitTorrent, eDonkey, and eMule. The BitTorrent has a constant low 0.1 poisoning effect, independent of chunk number. The number of chunks in a file has a tremendous impact on poisoning effect in eDonkey. The poisoning effect of eDonkey increases quickly as more chunks are added in a file. It becomes saturated with θ = 1, after the file grows beyond 64 chunks. As expected, eMule has very similar poisoning effect as eDonkey when the file contains less than 53 chunks. When the file include more than 53 chunk, although Equation 4 suggests that eMule has less poisoning effect than eDonkey, both values are so close to maximum that the difference are undistinguishable.

6.2 Repeated Peer Selection on Poisoning Effect

In previous experiments, we assumed that the client randomly picks a peer (provider or decoy). Many P2P clients repeat the selection of the same peer providing the current chunk to provide the next chunk. We called this strategy *repeated peer selection* (RPS) in Section 4.2. We assume that the 90% of the time the current provider is available to provide the next file chunk. The use of RPS policy does not affect the poisoning effects in BitTorrent and eDonkey networks using the random peer selection. The simulation results on using the RPS policy on eMule system are reported in Fig.4.

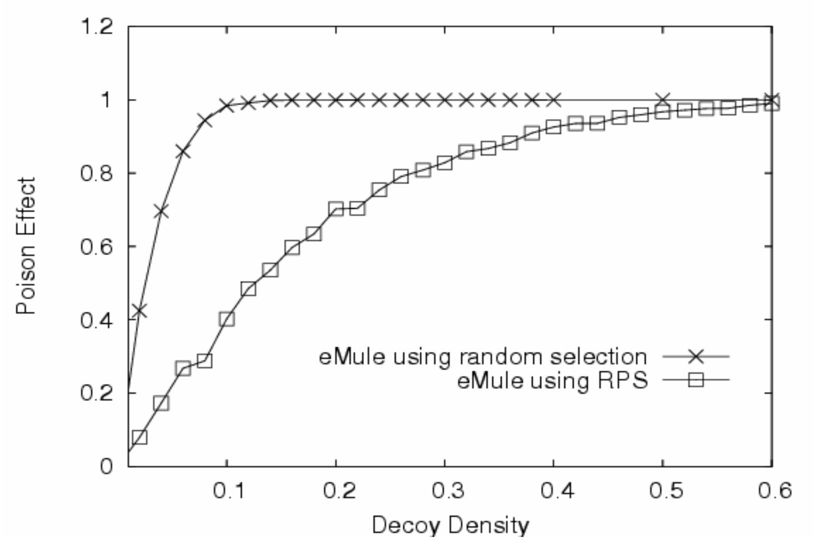
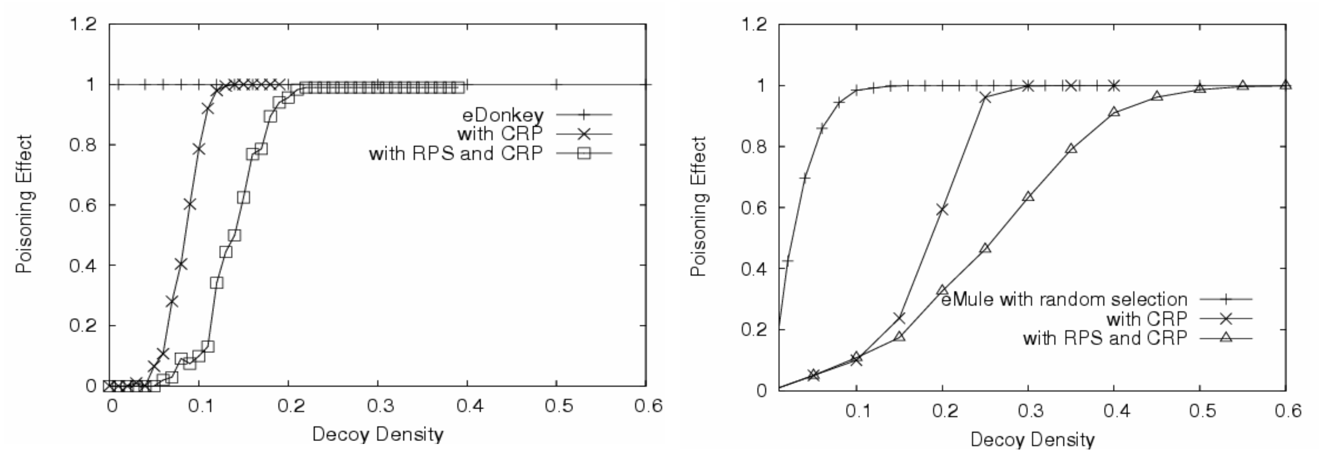


Figure 4 Effects of eMule using the repeated peer selection (RPS) policy, compared with using random peer selection policy in Fig.3(a).

Using the random peer selection, the eMule gets easily saturated to reach the maximum poisoning effect with 10% decoy density. Using the RPS policy, the growth slows down significantly. For example, the poisoning effect is 25% at 10% density. It takes more than 60% decoy density for eMule to reach the maximum poisoning effect. The message being conveyed here is that RPS policy will significantly increase a eMule network capability to resist content poisoning. However, the RPS performs about the same as using random policy in BitTorrent and eDonkey P2P networks. BitTorrent results are expected, since every chunk is independent. In the case of eDonkey, although poisoning effect are suppose to decrease compared to original protocol, the large file chunk number causes saturation even with very small decoy density.

6.3 Performance of using CRP Protocol

As discussed before, P2P network can apply the CRP protocol to increase the chance of receiving clean hashes from providers. This CRP protocol only affects the performance of eDonkey and eMule systems. Since the BitTorrent clients receive reliable hash values out-of-band, CRP is not applicable. Simulation results are shown in Fig.5 on augmenting the eDonkey and eMule systems with the CRP protocol and RPS policy.



a) eDonkey with CRP and RPS **(b) eMule with CRP and RPS**

Figure 5 Poisoning effects of eDonkey and eMule using CRP protocol and/or RPS policy

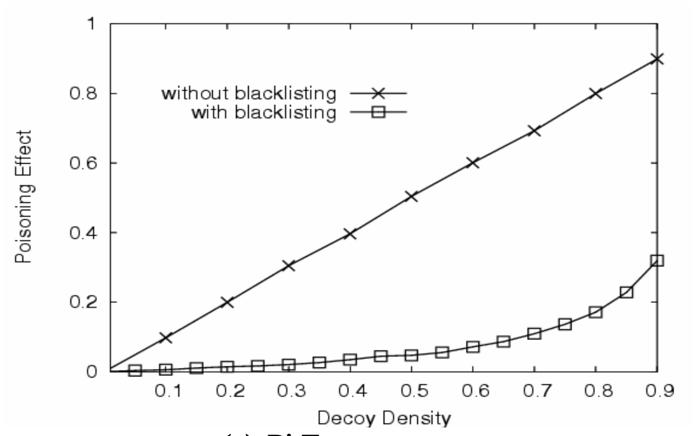
In eDonkey, the combined use of CRP and RPS reduces the poisoning effect as plotted in Fig. 5(a). Applying these two techniques, eDonkey starts to show some resistance to content poisoning. When 15% of the peers are decoys, poisoning effect is evaluated at 0.63. This is tolerable to many P2P file-sharing users. The combined use of RPS and CRP will lower even more poisoning effects in eMule system shown in Fig.5(b), compared with the eDonkey results shown in Fig.5(a). In these experiments, the poisoning effect can reach 90% for 40% decoy density. Both eDonkey and eMule become saturated with maximum poisoning of 1, once 60% of the peers are decoys.

An interesting observation is that the CRP protocol reduces poisoning effects with a few or no decoy used. At 10% decoy density, 10% poisoning is observed for “eMule with CRP” and “eMule with RPS and CRP”, significantly lower than that of “eMule with RPS”. As decoy density increases, eventually both curves with CRP surpass the curve for “eMule with RPS”. This phenomenon is explained by the majority voting mechanism in CRP. When the decoy density is low, majority voting favors the regular providers. As decoy density increases, the voting results are more likely to favor decoys. Eventually when there are more decoys than providers in the system, decoys are more likely to be selected, thus the increase of poisoning effect.

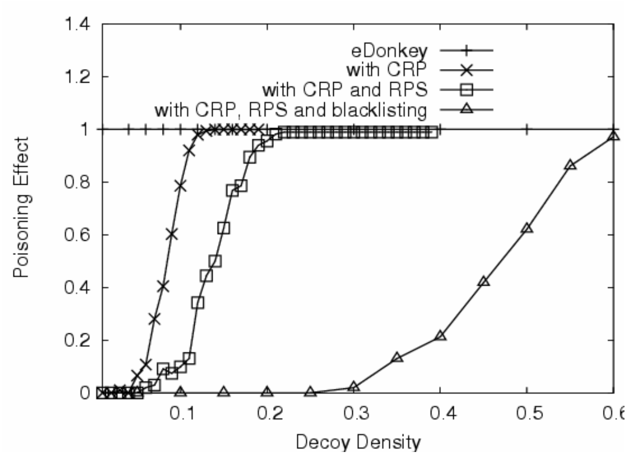
6.4 Effects of using the Decoy Blacklist

Some P2P file-sharing software has already included a rudimentary reputation system called *blacklist*. A blacklist contains peer identities suspected or proven as decoys. In this section, we reveal the application effects of using blacklist in BitTorrent, eDonkey and eMule systems through simulations. For simplicity, we assume 5% probability for the blacklist to include a clean provider and 95% probability to include a decoy. We also assume that the file contains 1,000 chunks. With the support of a blacklist, a client prefers to select a peer that is not on the blacklist. However, due to blacklist error, such a peer could be a decoy. On the other hand, blacklist error will also affect the majority-voting in the CRP protocol. This is due to the fact some providers are blacklisted and thus excluded from the peer selection.

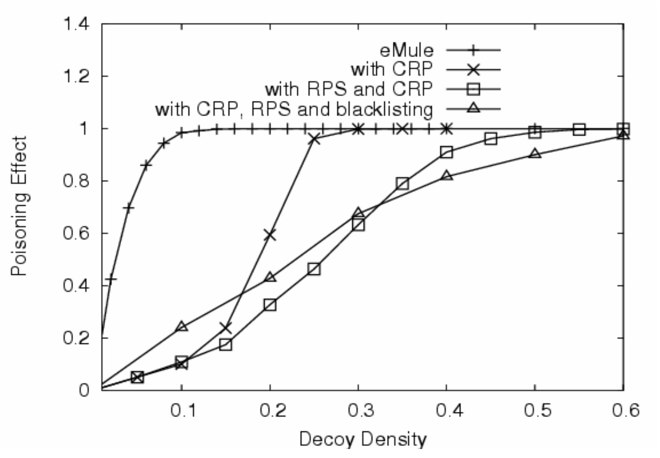
As shown in Fig.6(a), blacklisting is rather effective to lower poisoning effect in the BitTorrent system. When 90% of the peers are decoys, the poisoning effect has dropped from 90% to 33%. Considering the fact that most P2P file-sharing software runs at background and users are more patient towards P2P downloading, such low level of content poisoning is not effective. In fig.6(b), the lowest curve corresponds to the joint use of the CRP, RPS, and blacklisting on eDonkey. We observe that using blacklist reduce the poisoning effect to zero when there are less than 25% decoys. It requires more than 60% decoys to restore the poisoning effect to saturation. Compared with the modified eDonkey protocol with CRP, content poisoning is much less effective.



(a) BitTorrent system



(b) eDonkey system



(c) eMule system

Figure 6 Poisoning effects of application of blacklisting along with the use of the CRP protocol and RPS policy on three P2P file-sharing systems

As shown in Fig.6(c), the eMule system preserves more poisoning effects with a slower reduction than that of eDonkey, when CRP, RPS, and blacklisting are applied. This observation is contrary to the original design goal of eMule protocol by providing verification at the part level. Because the part hashset is distributed in the P2P network, it is subject to content poisoning as a common file chunk. If the part hashset is poisoned, not only the entire file is poisoned, but also no detection of such poisoning by the client at all. In our simulation, poisoning the part hashset has preserved the poisoning effect level in using the transformed eMule system.

7. Decoy Deployment Strategies and Limitations

In previous sections, we analyze various content poisoning techniques from the P2P users' perspective. In this section, we discuss options for content owners to improve poisoning effect.

7.1 Guidelines of Decoy Deployment

It is counterproductive to set up ultra fast decoys and advertise them as fast clean providers. As reported in [9] , P2P clients are unlikely to select the fastest peers for download, because of worry about being hooked by decoys. As the name decoy suggests, the best strategy is to blend in with other peers so that P2P users cannot easily identify decoys from clean peers. Different P2P file-sharing systems have very different levels of resistance to content poisoning.

P2P system can take effective steps to become more poison resistant. Content owners need to explore these differences in the targeted system deployed. This has motivated us to put forward the adaptive approach to content poisoning. Figure 7 reports decoy requirements of three P2P file-sharing systems under their original protocols plus the applications of the CRP protocol, RPS policy, and blacklisting.

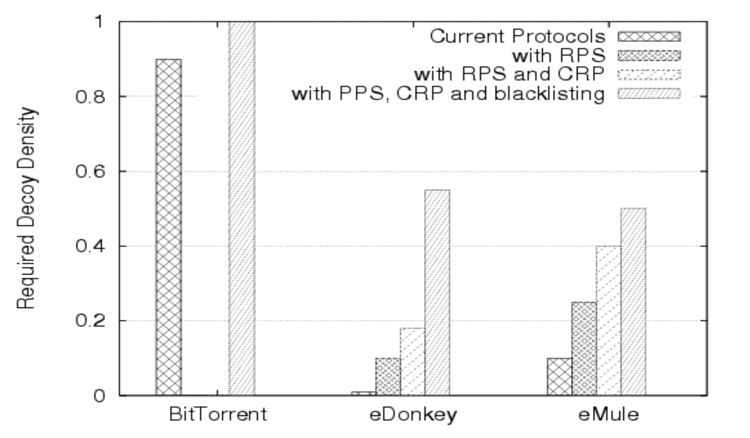


Figure 7 Decoy requirements of three P2P file-sharing systems under the application of CRP protocol, RPS policy, and blacklisting. To achieve a 90% poisoning effect, original BitTorrent protocol requires 90% decoy density, compared with less than 1% in eDonkey and 10% in eMule.

In original protocols, to achieve a 90% poisoning effect, BitTorent requires 90% decoy density, compared with less than 1% and 10% in using eDonkey and eMule, respectively. Applying all three techniques, BitTorrent requires to convert all peers to decoys, which may not be feasible due to the resistance from illegal providers. The eDonkey requires a decoy density of 55%

and eMule require only 50%. These findings agree with those reported in [4] Because content owner cost is directly related to decoy numbers, such high decoy density in BitTorrent implies very high cost to content owners.

Furthermore, blacklisting does not work on BiTorrent network either. In eDonkey and eMule, content poisoning have satisfactory results with limited decoy density. The result that eDonkey is more resistant than eMule is rather unexpected, because it contradicts the original design goal of eMule protocol. We reason with poisoned file chunks, the poisoned part-level hashsets have contributed greatly in eMule to make poisoning effectively under control.

7.2 Cost of Decoy Deployment

The cost of the content owner is directly related to the number of decoys deployed and the cost of each decoy. Previous sections focus on the number of decoys; in this section we analyze the bandwidth cost of each decoy. Based on current technology, the hardware cost for a decoy is ignorable; therefore the majority cost of a decoy is the network bandwidth it consumes. Most P2P users are connected to the Internet via broadband connections, because dial-up is not practical for downloading files with hundreds of Megabytes.

In our experiment, we assumed that the clients as well as providers are connected to their ISP via ADSL connections, with a maximum upload bandwidth of 256 Kbps. This parameter sets an upper bound on the bandwidth requirement. Many P2P client software supports setting lower upload bandwidth, but are usually turned off by users to encourage sharing. Figure 8 shows the average bandwidth consumption for each decoy reduces as poisoning effect increases. Particularly, BitTorrent decoy bandwidth consumption drops much faster than eDonkey and eMule. Decoys in BitTorrent are less likely to be selected because of its resistance to content poisoning. For eDonkey and eMule, bandwidth consumption for each decoy is close to 12 Kbps, when poisoning effect approaches 90%. In reality, popular files have many downloading clients, and each client can become providers. This will substantially increase the bandwidth consumption for each decoy.

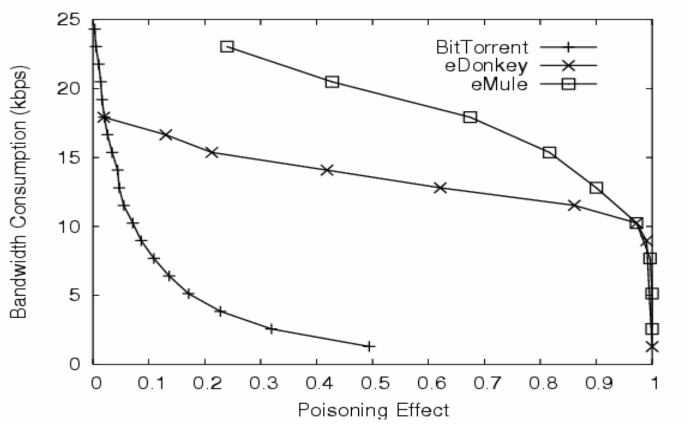


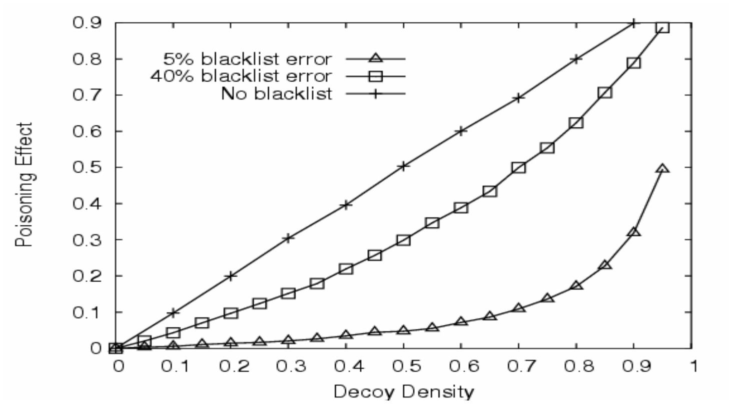
Figure 8 Average bandwidth consumption of each decoy under various poisoning requirements

7.3 Impact of Decoy Blacklist Error

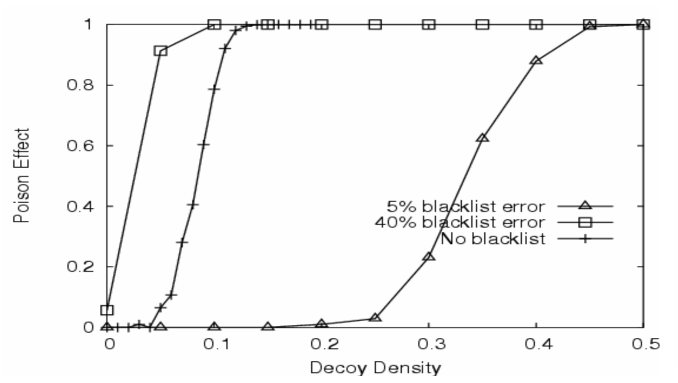
The static nature of blacklisting suggests a countermeasure by the poisoner. While old decoys might be exposed, new decoys are unlikely to be quickly identified by manual blacklisting. Therefore, the poisoner can periodically inject new decoys to replace old ones that are already blacklisted, or even change the address of old decoys so that IP address-based blacklist will not be easy to detect. This periodic update of blacklist may inject some errors. Fig.9 reveals that higher blacklisting error increases the poisoning effect, effectively.

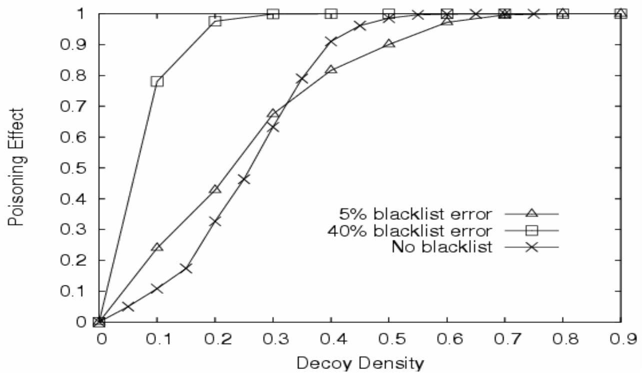
In BitTorrent, increasing blacklist error considerably to 40%, the poisoning effect is similar to that of no blacklist. However, it is still difficult for poisoners, since achieving 90% poisoning effect requires too many decoys. For this reason, we do not encourage the use of poisoning decoys in BitTorrent system. In eDonkey, raising blacklist error creates higher poisoning effect. When blacklisting errors are low (5%), 55% decoy density can effectively stop illegal file distribution.

As blacklisting error increases to higher levels, the decoy density requirements drop quickly. Similarly in eMule, high blacklisting errors led to lower decoy density in order to stop illegal distribution of copyrighted files. As shown in Fig.9(c), only 50% decoy is needed when blacklist error is 5%. In both eDonkey and eMule, high blacklisting error (40%) leads to poisoning effect even higher than those without the list. This shows that unreliable blacklist has a reverse effect.



(a) BitTorrent





(b) eDonkey **(c) eMule**

Figure 9 Impact of decoy blacklisting error on the poisoning effects in three P2P systems

7.4 Limitations of P2P Content Poisoning

After the theoretical analysis and extensive experimentation, we have identified three practical limitations on using content poisoning for copyright protection in P2P networks.

A. High cost to monitor and maintain all protected Files:

The content owner must constantly monitor the P2P networks to identify pirated content. Once such contents are found, decoys must be injected into P2P systems to poison the specific file. For example, a software company should watch for its installation CD files being shared on the P2P networks. It must also scan for many formats of compressed file of the same CD. The fact that content poisoning is file based suggests very high maintenance cost to the content owner. The owner must look for all owned contents, as well as any variations of those contents.

B. A technical battle to stop piracy with limited social support:

Content poisoning is a statistical countermeasure. Although this technology targets those users that attempt illegal download of the file, there is no guarantee that every illegal download will fail. The goal of content poisoning is not to frustrate every user. Rather, it seeks to discourage user from using free P2P network for illegal exchanges. When more users are discouraged, the pirated file will be less popular. By reducing downloading user number, content poisoning achieves its goal of copyright protection.

C. Need to combine content poisoning with index poisoning:

Content poisoning proves to be ineffective in BitTorrent network, as demonstrated by our model and experiments. Copyright protection in BitTorrent must seek index poisoning to solve the problem. Unlike content poisoning, index poisoning provides false information when P2P user is making inquires to the file. Instead of pointing to real peers that are sharing the file, decoys point clients to completely irrelevant files and/or hosts. In this process, most available index-poisoning schemes victimize the innocent hosts [22] .Therefore, many researchers consider index poisoning a kind of network attacks. We propose to solve this problem in further effort in Section 8.2.

8. Conclusions and Suggestions

Peer-to-peer networks are becoming popular and economic means to share digital contents on the Internet. However, this open media becomes also a crime bed to distribute copyrighted files or to steal protected information from peer resources. We suggest an adaptive content poisoning scheme to solve this problem. The contributions of this paper are summarized below in four technical aspects. Then we identify three interesting topics for further research challenges.

8.1 Summary of Contributions

Content owners want to use content poisoning to protect their copyrighted files in the P2P systems, while the technique is resisted by peer clients. We find that the underlying file chunking protocol used in a P2P file sharing system plays a critical role on the effectiveness of content poisoning. This is evidenced by the new poisoning effect models presented and by the experimental results reported in this paper.

A. *Relative merits of three P2P network systems for copyright protection*:

BitTorrent, eDonkey, and eMule apply different chunking and hashing protocols. Their capabilities to resist content poisoning are revealed. BitTorrent can verify each file chunk and thus highly poison-resistant. We do not recommend the use of content poisoning on BitTorrent network. However, we suggest to use index poisoning to point clinet to wrong files or even non-existing hosts to achieve copyright protection goals. The eDonkey and eMule systems are shown suitable to apply content poisoning using a small number of decoys. The eMule system performs better than the eDonkey system with only a few decoys.

B. *Adaptive poisoning for protected file transfer in Improved P2P Networks*:

Adaptive poisoning appeals to both eDonkey and eMule systems. The eDonkey clients do not process the hash of each chunk. They are not able to detect poisoned file chunks, nor can they identify the poisoned chunk after the entire file has been downloaded. The eMule protocol provides a part-hashset exchange mechanism in the P2P network. This mechanism makes the system more resistant to content poisoning. However, poisoned hashset prevents eMule client from completely download the file. This has limited further improvement of eMule protocol.

C. *Modeling poisoning effects of various file chunking protocols applied*

We derive mathematical models of poisoning effect for three P2P networks. These three models also cover other P2P protocols that have similar file chunking and hashing schemes. Experiments prove the validity of these models. Based on our analysis, content owner gain theoretical understanding of the expected performance of content poisoning.

D. *New CRP protocol, blacklisting of decoys, and repeated peer selection:*

We explored a list of techniques that can increase P2P system's resistance to content poisoning, including *repeated peer selection* (RPS), *challenge response protocol* (CRP), and blacklisting. These techniques can be use to control the degree of content poisoning to yield cost-effective implementations. Our experiments suggest that content poisoning is rather ineffective in BitTorrent. However, eDonkey and eMule are both sensitive to these tuning techniques. These techniques offers the option to perform fine-tuning of poisoning performance in eDonkey and eMule P2P networks.

8.2 Suggestions for Further Research

We suggest three tasks for further research challenges in P2P copyright protection. First, we need to investigate the poisoning effects under variable number of decoys used. Second, we need to explore index poisoning techniques to protect copyrighted file. Third, we suggest to integrate DRM and content poisoning techniques.

E. Content poisoning using variable number of decoys, dynamically:

This paper analyses the effects of content poisoning on different P2P file chunking schemes under a static condition of using a fixed number of decoys. However, in protocols that a client is not able to obtain reliable chunk hashes, poisoned file chunks will propagate to other clients and legitimate peers will become decoys unknowingly. Such dynamic change of decoy density and its impact on content poisoning are yet to be explored by further research.

F. Index poisoning to disrupt client query of file index:

Content poisoning seeks to disrupt illegal download after the client acquires the file index. Another viable approach is index poisoning at query stage. This will disrupt client query of the file index. This approach sometime is regarded as network attack, since it causes unintended traffic flow and potentially bring down the entire P2P network. We will look into possible ways of index poisoning to protect copyrighted content in a P2P network, without attacking the infrastructure. We expect the index poisoning to have very different impact on all P2P networks.

G. Integrating DRM with content poisoning and reputation systems in P2P systems:

DRM systems are quite expensive to deploy in a P2P environment. They are yet to be proven effective to stop all sorts of abuses in copyright violations. In P2P network environment, the DRM approach is particularly difficult due to lack of control of peer behavior. The peer collusion problem is difficult to solve. The reputation systems are suggested to establish trust among the peers. We feel that content poisoning can play a complementary role on top of using DRM and reputation systems in P2P content delivery services. To integrate the three ideas in one seamless copyright-protection system is a non-trivial job. Extensive research and prototyping are needed to prove the concept. This area is wide open for all P2P researchers. The works by Gedik and Liu [12], Mei, etc al [28, 41], Yurkewych, et al [40], and Ridriguez, et al [31] are rather inspiring in these directions.

Acknowledgements: *This work was supported by NSF ITR Grant ACI-0325409 at USC Internet and Grid Research Laboratory. We appreciate valuable suggestions from our team members.*

References

[1] J. E. Bailes and G. F. Templeton, "Managing P2P Security," *Comm. ACM* 47, 9 (Sep. 2004), 95-98.

[2] D. Bernstein, Z. Feng, B. Levine, and S. Zilberstein, "Adaptive Peer Selection," *The 2nd International Workshop on Peer-to-Peer Systems*, 2003.

[3] CacheLogic. (2005), "First-ever, Real-time Traffic Analysis of File Formats Crossing Peer-to-Peer", released by CacheLogic. http://www.cachelogic.com/news/pr090805.php?cnn=yes

[4] A. K. Choudhury, N. F. Maxemchuk, S. Paul, and H. G. Schulzrinne, "Copyright Protection for Electronic Publishing over Computer Networks," *IEEE Trans. on Networking*, vol. 9, pp. 12-20, 1995.

[5] N. Christin, A. S. Weigend, and J. Chuang, "Content Availability, Pollution and Poisoning in File Sharing Peer-to-Peer Networks," *Proceedings of the 6th ACM Conference on Electronic Commerce*. New York, ACM Press, 2005, pp. 68–77.

[6] E. Damiani, D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante, "A Reputation-based Approach for Choosing Reliable Resources in Peer-to-Peer Networks," *CCS '02: Proceedings of the 9th ACM Conf. on Computer and communications security*. N. Y., ACM Press, 2002, pp. 207–216.

[7] P. Dewan and P. Dasgupta, “Securing P2P Networks Using Peer Reputations: Is There a Silver Bullet?” *Consumer Communications and Networking Conference, 2005. CCNC. 2005*, pp. 30–36.

[8] R. Dingledine, M.J. Freedman, and D. Molnar, “Accountability Measures for Peer-to-Peer Systems,” *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, O'Reilly Publishers, 2000.

[9] D. Dumitriu, E. Knightly, A. Kuzmanovic, I. Stoica, and W. Zwaenepoel, “Denial-of-Service Resilience in Peer-to-Peer File Sharing Systems,” *Proc. of ACM Int’l Conf. on Measurement and modeling of computer systems*. New York, ACM Press, 2005, pp. 38–49.

[10] Faq: ed2k-kademlia. http://www.amule.org/wiki/index.php/FAQ_ed2k

[11] M. Fetscherin and M. Schmid, "Comparing the Usage of Digital Rights Management Systems in the Music, Film, and Print Industry", *Proc. of the 5th International Conf. on Electronic commerce*, 2003

[12] B. Gedik and L. Liu, "A Scalable Peer-to-Peer Architecture for Distributed Information Monitoring Applications," *IEEE Trans. on Computers*, pp. 767-782, June 2005.

[13] M. Gupta, P. Judge, and M. Ammar, “A Reputation System for Peer-to-peer Networks,” *Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*. New York, ACM Press, 2003, pp. 144–152.

[14] J. C. Hale and G. W. Manes, “Method to Inhibit the Identification and Retrieval of Proprietary Media via Automated Search Engines Utilized in Association with Computer Compatible Communications Network,” U.S. Patent 6,732,180, 2004.

[15] M. Hofmann and I. Beaumont, *Content Networking, Architecture, Protocols, and Practice*, Morgan Kaufmann, Elsevier, S.F. 2005.

[16] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, “The EigenTrust Algorithm for Reputation Management in P2P Networks,” in *WWW ’03: Proceedings of the 12th international conference on World Wide Web*. New York, ACM Press, 2003, pp. 640–651.

[17] S. Knox, D. O’Cearuil, N. S. Holland, and L. Skrba. “Technology Survey: BitTorrent.” http://ntrg.cs.tcd.ie/undergrad/4ba2.05/group4/index.html

[18] B. Krishnamurthy, C. Wills, and Y. Zhang, “On the Use and Performance of Content Distribution Networks,” *Proc. of SIGCOMM IMW*, Nov. 2001.

[19] Y. Kulbak and D. Bickson, “The eMule Protocol Specification,” Hebrew University *Technical Report*, Tech. Rep. TR-2005-03, Jan. 2005.

[20] J. Lee, “An End-user Perspective on File-sharing Systems,” *Comm. of Association of Computing Machinery*, ACM Press, pp. 49–53, 2003.

[21] J. Liang, R. Kumar, Y. Xi, and K. Ross, “Pollution in P2P File-Sharing systems,” *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, 2005, pp. 1174–1185.

[22] J. Liang, N. Naoumov, and K. W. Ross, “The Index Poisoning Attack in P2P File-sharing Systems,” *Infocom*, 2006.

[23] Q. Liu, R. Safavi-Naini and N. Sheppard, "Digital Rights Management for Content Distribution", *Australasian Information Security Workshop* 2003.

[24] D. Manini, R. Gaeta, and M. Sereno, “Performance Modeling of P2P File Sharing Applications,” *Techniques, Methodologies and Tools for Performance Evaluation of Complex Systems, (FIRB-Perf 2005).* 2005, pp. 34–43.

[25] A. Mei, L.V. Mancini, ans S. Jajodia, “Secure Dynamic Fragmment and Replica Allocation in Large-Scale Distributred File Systems”, *IEEE Trans. Parallel and Distributed Systems,* Vol. 14, No.9, Sept. 2003

[26] N. Mook, “BetaNews | P2P Flooder Overpeer Ceases Operation,” Dec. 10, 2005 http://www.betanews.com/article/P2P_Flooder_Overpeer_Ceases_Operation/1134249644

[27] O’Reilly Radar Blog, “HBO Attacking BitTorrent.”,October 2005. http://radar.oreilly.com/archives/2005/10/hbo_attacking_bittorrent.html

[28] G. Pallis and A. Vakali, 2006. “Insight and Perspectives for Content Delivery Networks,” *Comm.of The ACM, Vol.* 49, No.1 (Jan. 2006), pp.101-106

[29] G. Pierre, M. van Steen, and A. S. Tanenbaum, "Dynamically Selecting Optimal Distribution Strategies for Web Documents," *IEEE Trans. on Computers*, vol. 51, pp. 637-651, 2002.

[30] J. A. Pouwelse, P. Garbacki, D. H. Epema, and H. J. Sips, “The BitTorrent P2P File-sharing System: Measurements and Analysis,” *4th In’ll Workshop on Peer-to-Peer Systems* (IPTPS’05), 2005.

[31] P. Rodriguez, S. Tan and C. Gkantsidis, “On the Feasibility of Commercial Legal P2P Content Distribution, ” *SIGCOMM Comput. Commun. Rev.*36, 1 (Jan. 2006), pp.75-78.

[32] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy, “An analysis of Internet Content Delivery Systems,” *SIGOPS Operating System Review*, vol. 36, no. SI, pp. 315–327, 2002.

[33] Slashdot Discussion Archive, “Can Poisoning Peer to Peer Networks Work ?”, Sept. 2002, http://slashdot.org/comments.pl?sid=39198&threshold=1&commentsort=0&tid=95&mode=thread&cid=4188551

[34] S. Sen and J. Wang, “Analyzing Peer-to-peer Traffic Across Large Networks,” *IEEE/ACM Trans. Networking*, vol. 12, no. 2, pp. 219–232, 2004.

[35] R. Steinmetz and K. Wehrle, *Peer-to-Peer Systems and Applications*. Springer, 2005.

[36] K. Thompson, G. Miller and R. Wilder (1997), “Wide Area Internet Traffic Patterns and Characteristics,” *IEEE Network Magazine*, Nov. 1997

[37] K. Walsh and E. G. Sirer, “Fighting Peer-to-Peer SPAM and Decoys with Object Reputation,” *P2PECON ’05: Proc. of the 2005 ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems*. New York, ACM Press, 2005, pp. 138–143.

[38] C. Wu and B. Li, “Optimal Peer Selection for Minimum-Delay Peer-to-Peer Streaming with Rateless Codes,” *Proc. of the ACM Workshop on Advances in Peer-to-Peer Multimedia Streaming*. New York, ACM Press, 2005, pp. 69–78.

[39] L. Xiao, Y. Liu and L. M. Ni, "Improving Unstructured Peer-to-Peer Systems by Adaptive Connection Establishment," *IEEE Trans. on Computers*, Sept., 2005, pp. 1091-1103.

[40] M. Yurkewych, B. N. Levine, and A. L. Rosenberg, "On the cost-ineffectiveness of redundancy in commercial P2P computing," *Proceedings of the 12th ACM Conference on Computer and Communications Security*. Alexandria, VA, ACM Press, 2005.

[41] G. Zanin, L. V. Mancini, and A. Mei, "Towards a Secure Dynamic Allocation of Files in Large Scale Distributed File Systems", *Proceedings of the IEEE Workshop in Hot Topics in Peer-to-Peer Computing* (HOT-P2P 2004), 2004.

[42] R. Zhou and K. Hwang, “PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing”, *IEEE Trans. Parallel and Distributed Systems*, to appear 2006.

Biographical Sketches:

Xiaosong Lou received the B.S. degree in Electronic Engineering from Shanghai Jiaotong University in 1994. He has worked in several information technology companies as system and application engineer for 10 years. In 2005, he received his M. S. degree in Computer Engineering from the University of Southern California. Currently, he is a Ph.D. candidate in the USC Computer Engineering program. His research interest covers the areas of peer-to-peer and Grid computing, poisoning detection, distributed content delivery, and semantic networks.. He can be reached via the email address: xlou@usc.edu

Kai Hwang is a Professor of Electrical Engineering and Computer Science and Director of Internet and Grid Research Laboratory at USC. He received the Ph.D. degree from the University of California, Berkeley. An IEEE Fellow, he specializes in computer architecture, parallel processing, Internet and wireless security, Grid and cluster computing, and distributed computing systems.

Dr. Hwang is the founding Editor-in-Chief of the *Journal of Parallel and Distributed Computing* published by Elsevier. He is also on the editorial board of *IEEE Transactions on Parallel and Distributed Systems.* He has published over 200 original scientific papers and 7 research or textbooks. His latest books, *Scalable Parallel Computing* and *Advanced Computer Architecture* have been translated into 4 languages. Presently, he leads the GridSec project at USC in developing distributed defense systems against worms and DDoS attacks for trusted Grid, P2P, and Internet computing. Contact him at kaihwang@usc.edu or visit http://GridSec.usc.edu/Hwang.html for details.