Proactive Content Poisoning To Prevent Collusive Piracy in P2P File Sharing

Xiaosong Lou, *Student Member IEEE* and **Kai Hwang,** *Fellow IEEE Computer Society*

Abstract: Today's *peer-to-peer* (P2P) networks are grossly abused by Illegal distributions of music, games, video streams, and popular software. These abuses have resulted in heavy financial loss in media and content industry. *Collusive piracy* is the main source of intellectual property violations within the boundary of P2P networks. This problem is resulted from paid clients (colluders) illegally sharing copyrighted content files with unpaid clients (pirates). Such an on-line piracy has hindered the use of open P2P networks for commercial content delivery. We propose a proactive poisoning scheme to stop colluders and pirates from working together in alleged copyright infringements in P2P file sharing. The basic idea is to detect pirates with identity-based signatures and time-stamped tokens. Then we stop collusive piracy without hurting legitimate P2P clients.

We developed a new *peer authorization protocol* (PAP) to distinguish pirates from legitimate clients. Detected pirates will receive poisoned chunks in repeated attempts. A reputation-based mechanism is developed to detect colluders. The system does not slow down legal download from paid clients. The pirates are severely penalized with no chance to download successfully in finite time. Based on simulation results, we find 99.9% success rate in preventing piracy on file-level hashing networks like Gnutella, KaZaA,Area, LimeWire, etc. Our protection scheme achieved 85-98% prevention rate on part-level hashing networks like eMuel, Shareaz, eDonkey, Morpheus, etc. Our new scheme enables P2P technology for building a new generation of *content delivery networks* (CDNs). These P2P-based CDNs provide faster delivery speed, higher content availability, and cost-effectiveness than using conventional CDNs built with huge network of surrogate servers.

Index Terms— Peer-to-peer networks, content poisoning, content delivery networks, and network security.

1 INTRODUCTION

Peer-to-peer (P2P) file-sharing networks are most cost-effective in delivering large files to massive number of users [3], [33]. Unfortunately, on-line piracy has hindered the legal and commercial use of P2P technology. The main sources of illegal file sharing are peers who ignore copyright laws and collude with pirates. To solve this peer collusion problem, we propose a proactive copyright-compliant system for protecting legalized P2P content delivery.

Our goal is to stop collusive piracy within the boundary of a P2P content delivery network. Our protection scheme works nicely in a P2P network environment. The scheme cannot stop randomized piracy in open Internet using Email attachment or any other means to spread copyrighted contents, illegally. Randomized piracy is beyond the scope of this study.

Traditional *content delivery networks (*CDN)[13][17][24] use a large number of surrogate content servers over many globally distributed WANs. The content distributors need to replicate or cache contents on many servers. The bandwidth demand and resources needed to maintain these CDNs are very expensive.

A P2P content network significantly reduces the distribution cost[27], since many content servers are eliminated and open networks are used. P2P networks improve the content availability, as any peer can serve as a content provider. P2P networks are desired to be scalable, because more peers or providers lead to faster content delivery.

We use *identity-based signatures* (IBS) [4] to secure file indices. IBS offers the same level of security as PKI-based signatures with much less overhead. We apply discriminatory content poisoning against pirates. We focus on protection of decentralized P2P content networks. Protecting centralized P2P networks like Napster or mp3.com is much simpler than the scheme we proposed.

Our goal is to stop collusive piracy within the boundary of a P2P content delivery network. Our scheme cannot stop primary piracy in open Internet using Email attachment or any other means to spread copyrighted contents, illegally. In particular, our scheme appeals to protect perishable or large-scale contents that diminish in value as time elapses. Our system stops abuses within the P2P network, exclusively.

Honest or legitimate *clients* are those that comply with the copyright law not to share contents freely. *Pirates* are peers attempting to download some content file without paying or authorization. The *colluders* are those paid clients who share the contents with pirates. Pirates and colluders coexist with the law-abiding clients.

Content poisoning is realized by deliberate falsification of the file requested by pirate. The media industry backed by RIAA (*Record Industry Association of America*) and MPAA (*Motion Picture Association of America*) has applied unscreened brutal-force content poisoning to deter piracy in open P2P file-sharing networks. However, their prevention results are mixed and controversial.

For example, the Overpeer [23] ceased operation in 2005 for ineffective universal poisoning of all peer clients. MediaDefender[1] has caused a heated controversy over P2P user community, when their operations disrupted many P2P services. The eDonkey overlay was ordered by court to shut down in 2006 [24] due to incontrollable piracy activities. The popular BitTorrent and Freenet networks are still facing many lawsuits against their content distribution operations [22].

The media industry applies content poisoning to all peers in order to shutdown all P2P file-sharing services by brute force. In contrast, our scheme detects unpaid

pirates and use discriminatory content poisoning to deter on-line piracy. Legitimate clients can still enjoy the flexibility and convenience provided by open P2P networks. Our scheme stops pirates from illegal download of copyrighted files, even at the presence of many colluding peers. We developed a reputation-based method to detect peer collusion in piracy process.

A copyright protected P2P network should benefit both media industry and Internet user communities [6]. Our work leads to the development of a new generation of CDNs based on P2P technology. Table 1 lists important symbols and notations used to benefit our readers. These terms are used to secure file indices; generate access tokens; quantify poisoning effects, collusion prevention, and to define the performance metrics.

TABLE 1. PARAMETERS AND NOTATIONS USED IN PAPER

Term, Symbol	Brief Definition
Access token, T	A short-life token for access control of content file
Time stamp, t_s	Used in securing file index /query/requests
User address, p	User endpoint log-in address observed by the agent
File index, ϕ	Pointer to access the requested content file
Clean file size, f	Original file size in bytes without poisoning
Download file, d	Actual bytes downloaded, possibly poisoned ($d \geq f$)
Poisoning rate, δ	Probability of receiving a poisoned chunk by pirates
Chunk number, m	Number of chunks in a single content file
Collusion rate, ε	Percentage of paid peers acting as colluders
Piracy rate, r	Percentage of pirates detected in a P2P network
Download times, T_c and T_p	Expected times to download a clean file by a paid client and by a detected pirate, respectively
Tolerance, θ	Maximum download time tolerable by peers
Success rate, β	Probability of detecting and preventing a pirate

We focus on finding solution of collusive piracy within the scope of a P2P network. Inter-network piracy between unprotected P2P networks is a much more complex security problem. That compound problem is not within this study. Our main purpose is to stop colluders from releasing content files freely and to abort pirate effort from accumulating clean chunks. There are many other forms of on-line or off-line piracy that are beyond the scope of this study.

For examples, our protection scheme does not work on a private or enclosed network formed by pirate hosts exclusively. We did not solve the piracy problems using email attachments; FTP download directly between colluders; or using replicated CD or DVD disks. At present, these direct point-to-point copyright violation problems are most handled by *digital rights management* (DRM) techniques [21], even the protection results are not considered satisfactory [10], as many hackers have post DRM-cracks on Internet.

2 RELATED WORK AND OUR APPROACH

We review related work on copyrighted P2P content delivery. Then we identify our unique contributions.

2.1 Related Work

A P2P network does not require many expensive servers to deliver contents. Instead, contents are distributed and shared among the peers. P2P networks improve from conventional CDNs in content availability and system scalability [3]. Many performance and security issues in P2P networks have been studied, such as in [5], [12], [30], [31], [32]. Digital content protection in P2P networks was also studied in [15], [29].

Electronic publishing was hindered by the rapid growth of copyright violations [14], [26]. The major source of illegal P2P content distribution lies in peer collusion to share copyrighted content with other peers or pirates. Kalker et al [16] proposed a copyrighted music distribution over a P2P network. However, the system is ineffective when colluders are undetected.

Digital watermarking is often considered for digital copyright protection [19]. Digital watermarking is injected to content file so that when a pirated copy is discovered, authorities can find the origin of piracy via a unique watermark in each copy. In a P2P network, all peers are sharing exactly the same file (if not poisoned), which effectively defeats the purpose of watermarking. Thus, watermarking is not a suitable technology for P2P file-sharing.

By subdividing a large file into small chunks, P2P content delivery allows a peer to download multiple chunks from different sources. File chunking increases the availability and shortens the download time. Based on file chunking and hashing protocols built in popular open P2P networks, we classify them into three network families in Table 2. P2P networks in the same family have some common features. They are variants or descendants of their predecessors: *BitTorrent* [2], *Gnutella* [11], and *eMule* [18], respectively. These families are distinguished primarily by file chunking or hashing protocols applied. Presently, none of these P2P networks has built with satisfactory support for copyright protection.

TABLE 2 FILE CHUNKING, HASHING, POISONING, AND DOWNLOAD POLICIES IN P2P CONTENT NETWORKS

P2P Networks	*BiTorrent Family* [2]	*Gnutella Family* [11]	*eMule Family* [18]
Chunking Scheme	Divide files into fix sized chunks (256 KB), called pieces	Peers negotiate the chunk size at run time, 64 KB chunks by default.	Divide into 9500-KB parts, each has 53 (180 KB) chunks
Hash Distribution	SHA hashing at piece level, embedded in index file and distributed	SHA hashing applied to entire file to generate a unique file ID, no chunk-level hashing	MD-4 hashing at part level, peers exchange part-level hashset to detect corrupted contents.
Poison Resistance	Poison detected at piece level, each is handled independently	Poison detected only after down-loading the entire file, heavy overhead if poisoned	Poison detected at part level, works if part hashset is not poisoned,
Download Policy	Keep clean and discard poisoned pieces, repeated download until all chunks are clean	Repeat downloading entire file until all chunks become clean, the most time-consuming policy	Keep clean and discard poisoned parts, repeated download until all parts are clean
Example networks	BitTorrent, Snark, BitComet, BNBT, BitTyrant,Azureus, JTorrent, etc.	Gnutella, KaZaA, LimeWire, Phex, Freenet, BareShare, Swapper, Ares, etc	eMule, aMule, Shareaza, iMule, Morpheus, eDonkey, FastTrack, etc.

The BitTorrent family [2] applies the strongest hashing at individual *piece* or chunk level, which is most resistant to poisoning. The Gnutella family applies file-level hashing, which is easily poisoned. The eMule family [18] applies part-level hashing with fixed chunking. Our

analytical and experimental results show that the eMule family demonstrates a moderate level of resistance to content poisoning.

The ability to detect and identify poisoned chunks are different in three P2P network families. The BitTorrent family keeps clean chunks and discard poisoned chunks as each chunk is downloaded. The Gnutella family must download the entire file before any poisoned chunks can be detected. Because of the part-level hashing, the eMule family will either keep or discard an entire part, consists of 53 chunks.

2.2 Our Unique Contributions

Content poisoning is often treated as a security threat to P2P networks [7], [9]. To our best knowledge, using selective content poisoning to prevent collusive piracy has not been explored in the past. We offer the very first proactive poisoning approach to curtailing copyright violation in P2P networks. We make the following specific contributions towards P2P content delivery.

A. Distributed detection of colluders and pirates:

We develop a protocol that identifies a peer with its endpoint address. File index format is changed to incorporate this identity-based signature. A peer authentication protocol is developed to establish the legitimacy of a peer when it downloads and uploads the file. Using IBS, our system enables each peer to identify unauthorized peers or pirates without the need for communication with a central authority.

B. Proactive content poisoning of detected pirates:

Our protocol requires to send poisoned chunks to any detected pirate requesting a protected file. If all clients simply deny download request without poisoning, the pirates can still accumulate clean chunks from colluders that are willing to share. With poisoning, the pirates are forced to discard even clean chunks received. This will prolong their download time to a level beyond practical limit. Experiments show that it is unlikely that a pirate can download a clean copy of the file.

C. Containment of peer collusion to inspire piracy :

Our system is unique from any existing P2P copyright protection scheme in that we recognize that peer collusion is inevitable: a paid customer may intentionally collude with pirates; a pirate may also hack into client hosts and turn them into unwilling colluders. Our system is designed so that even with large number of colluders, a pirate will still suffer from intolerably long download time. We also present a random collusion detection mechanism to further enhance our system.

D. Trusted P2P platform for copyrighted content delivery:

Hardware investment for P2P content delivery is much lower than that required in any existing CDNs. Our system only use a few distribution agents to serve large number of clients. The system is highly scalable, robust to peer and link failures, and easily deployed in Gnutella, KaZaA, eMule networks, etc. All claimed advantages are backed by performance analysis and simulation results presented in subsequent sections.

3 Copyright-Protected P2P Networks

This section specifies the system architecture, client joining process, pirate poisoning mechanism, and colluder detection that we built in the newly proposed copyright-protection scheme for P2P content distribution in open network environment.

3.1 Trusted P2P Network Architecture

Our copyright-protected P2P network is depicted in Fig.1, conceptually. The network is built over a large number of peers. There are four types of peers coexist in the P2P network: *clients* (honest or legitimate peers), *colluders* (paid peers sharing contents with others without authorization), *distribution agents* (trusted peers operated by content owners for file distribution), and *pirates* (unpaid clients downloading content files illegally).

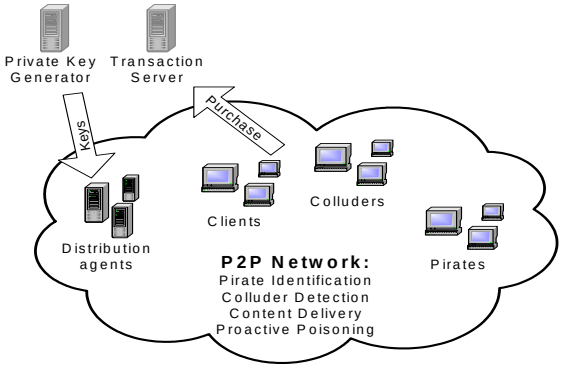


Figure 1 A protected P2P content delivery network, consisting of paid clients, colluders, pirates, and distribution agents. The design goal is to prevent pirates from downloading copyrighted files from colluders. Proactive poisoning is applied to pirates only without hurting paid clients. Only a handful of agents are used to handle the bootstrap and distribution of requested digital contents.

To join the system, clients submit the requests to a *transaction server* which handles purchasing and billing matters. A *private key generator* (PKG) is installed to generate private keys with *identity-based signatures* (IBS) for securing communication among the peers. The PKG has a similar role of a *certificate authority* (CA) in PKI services. The difference lies in that CA generates public keys distributed in IEEE 509 certificates, while PKG takes much lower overhead to generate private keys, which are used by local hosts.

The transaction server and PKG are only used initially when peers are joining the P2P network. With IBS, the communication between peers does not require explicit public key, because the identity of each party is used as the public key. In our system, file distribution and copyright protection are completely distributed.

Based on past experience, the number of peers sharing or requesting the same file at any point of time is around hundreds. Depending on the variation of the swamp size, only a handful of distribution agents is needed. For example, it is sufficient to use 10 PC-based distribution agents to handle a swamp size of 2,000 peers. These agents authorize peers to download and prevent unpaid peers from getting the same contents. Paid clients,

colluders, and pirates are all mixed up without labels.

Our copyright-protection network is designed to distinguish them automatically. Each client is assigned with a *bootstrap agent,* selected from one of the distribution agents, as its entry point. In current P2P networks, a peer can self-assert its username without verification. Therefore, we use peer endpoint address (IP address + port number) instead of username to identify a peer. A peer is considered fully connected if it is reachable via a listening port on its host.

We use the endpoint address of the listening port as a peer identity. For simplicity, we assume that each peer have a statistically configured listening port. Currently, most P2P users connect to the Internet via a home network. In such environments, statistically configuring the NAT device to forward incoming packets to a few P2P nodes is a norm. The constraint occurs when a large number of peers are behind a single NAT device.

Figure 2 depicts an example: a peer has IP address 192.168.0.2 leased from its local router. It is listening to port 5678 forward by the router. When communicating with the bootstrap agent, the peer announces its listening port number. The bootstrap agent calls an Observe() subroutine, which verifies that the same peer is indeed reachable via the claimed port, although its public IP address is actually 68.59.33.62. Hence the peer is identified by 68.59.33.62:5678.

The detail of Observe() is as follows: when a peer sends message to its bootstrap agent through outgoing port, agent attach a random number (nonce) in the reply. The agent then sends a message to the advertised listening port 68.59.33.62:5678, asking the peer to send back the nonce. If the peer replies correctly, then its endpoint is verified.

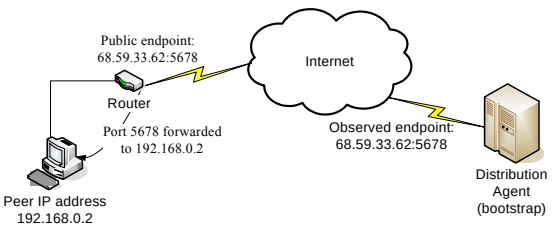


Figure 2 The bootstrap agent observes end-point address *p*= 68.59.33.62:5678 in a trust-enhanced P2P network

The endpoint address is used as peer's public key visible from outside. There is no need to encrypt the file body. This reduces the system overhead at peer level. All distribution agents, except the bootstrap agent, are hidden from clients. This design prevents a malicious node to blacklist or attack the distribution agents. Enabling peers behind NAT without static listening port require a "hole-punching" mechanism, and use its bootstrap agent to forward incoming requests. This level of implementation detail is implied.

3.2 Protection in Peer Joining Process

Figure 3 illustrates the process for a client to join a P2P network supported by a new *peer authorization protocol* (PAP). We will formally specify PAP in Section 4.3. Here, we first introduce the handshaking mechanisms used to protect the peer joining process. For a peer to join the network, it first logins to a transaction server to purchase the file. After transaction, the client receives a digital receipt containing the content title, client ID etc. This receipt is encrypted, only content owner and distribution agent can decrypt.

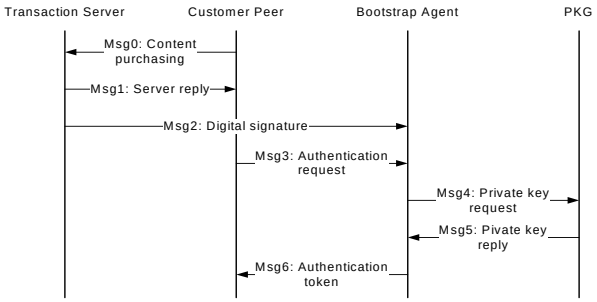


Figure 3 The protected peer joining process for copyrighted P2P content delivery. Seven messages are used to secure the communications among 4 parties involved.

The client receives the address of the bootstrap agent as its point of contact. The joining client authenticates with the bootstrap agent using the digital receipt. The session key assigned by the transaction server secures their communication. Since the bootstrap agent is setup by the content owner, it decrypts the receipt and verifies its authentication. The bootstrap agent requests a private key from PKG and constructs an authorization token, accordingly.

Let k be the private key of content owner and *id* be the identity of the content owner. We use $E_k(msg)$ to denote the encryption of message with key k. The $S_k(msg)$ denotes a digital signature of plaintext *msg* with key k. The client is identified by *userID* and the file by *fileID*.

Each legitimate peer has a valid token. The token is only valid for a short time so that a peer needs to refresh the token periodically. To ensure that peers not to share the content with pirates, the trusted P2P network modifies the file-index format to include a token and IBS peer signature. Peers use this secured file index in inquiries and download requests. Seven messages in the protected peer joining process are specified below.

Msg0: Content purchase request
Msg1: BootstrapAgentAddress,
E_k (digital_receipt, BootstrapAgent_session_key)
Msg2: Adding digital signature E_k (digital_receipt)
Msg3: Authentication request with
userID, fileID, E_k (digital_receipt)
Msg4: Private key request with
privateKeyRequest (observed peer address)
Msg5: PKG replies with privateKey
Msg6: Assign the authentication token to the client

Peers identify the pirates by checking the validity of extra signatures in file indices. The trusted P2P applies this protection to share clean contents exclusively among the peers, and use content poisoning techniques against the pirates. The identities of all agents, except the bootstrap agent, are hidden from the client. This design

ensures that a malicious node cannot blacklist or attack all distribution agents. Detected colluders cannot receive new token after the timestamp in current token expires.

3.3 Proactive Content Poisoning

We summarize in Table 3 the key protocols and mechanisms used to construct the trusted P2P system. In this approach, modified file index format enables pirate detection. PAP authorizes legitimate download privileges to clients. Content distributor applies content poisoning to disrupt illegal file distribution to unpaid clients. The system can be enhanced by randomized collusion detection among the peers.

In our system, a content file must be downloaded fully to be useful. Such a restraint is easily achievable by compressing and encrypting the file with a trivial password that are known to every peer. This encryption does not offer any protection of the content, except to package the entire file for distribution.

TABLE 3 MECHANISMS FOR COPYRIGHT PROTECTION

Mechanism	Protocol Requirements
Secure file Indexing	File index format is modified to include token and IBS signature
Peer Authorization Protocol (PAP)	Peer sends digital receipt to bootstrap agent and obtain an authorization token. The token must be refreshed periodically.
proactive Content Poisoning	The token and IBS signature check all download request and responses. Sending clean or poisoned contents, accordingly
Random Collusion Prevention	Distribution agents randomly recruit decoys to probe for colluders. Collusion reports are weighted against client trust rates

Figure 4 illustrates the proactive content poisoning mechanisms built in our enhanced P2P system. If a pirate sends download request to a distribution agent or a client, then by protocol definition it will receive poisoned file chunks. If the download request was sent to a colluder, then it will receive clean file chunks. If a pirate shares the file chunks with another pirate, then it could potentially spread the poison.

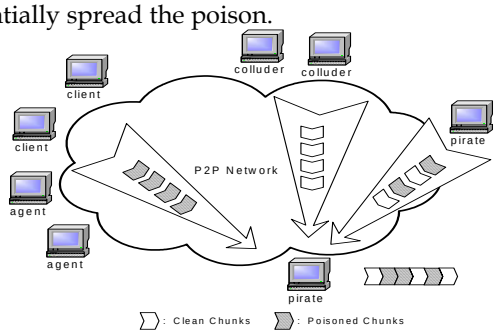


Figure 4 Proactive poisoning mechanism of the trusted P2P network, where clean chunks (white) and poisoned chunks (shaded) are mixed in a stream downloaded by a pirate, but legitimate clients receive only clean chunks.

Therefore, it is critical to send poisoned chunks to pirates, not simply denying their requests. Otherwise, even if all clients deny pirate's requests, the pirate still can assemble a clean copy from those colluders who have responded with clean chunks. With poisoning, we exploit the limited poison detection capability of P2P networks and force a pirate to discard the clean chunks downloaded with the poisoned chunks. The rationale behind such poisoning is that if a pirate keeps downloading corrupted file, the pirates will eventually give up the attempt out of frustration.

3.4 Randomized Colluder Detection

Although our system is designed to tolerate the presence of colluders in the network, we show in later sections that reducing number of colluders will improve system performance. Therefore, we introduce a reputation-based[8] colluder detection mechanism to secure our system from piracy.

As reported in our earlier work [34], gossip protocol and power nodes play a crucial role in speeding up the reputation aggregation process in a P2P network. Randomized gossiping can reach consensus among all peers in a distributed manner. This approach exploits massive concurrency among millions of active nodes in a very large P2P network. We design a simplified GossipTrust system to identify colluders in this paper.

The idea is to associate each {*peer, file*} pair with a *collusion rate.* The "0" rate means that the peer was never reported as a colluder. Otherwise, the peer is getting a collusion report of "1", meaning it has shared clean content with illegal download requesters. This collusion rate is accumulative like the way e-Bay collects peer's reputation scores. Figure 5 illustrates the collusion detection process.

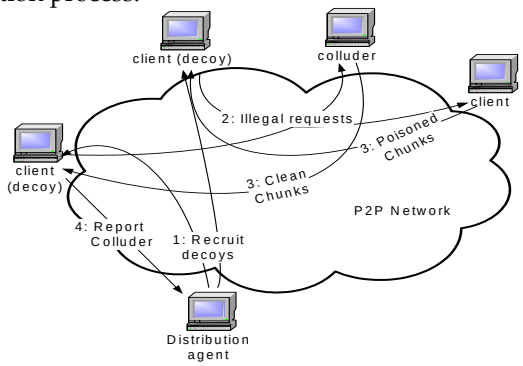


Figure 5. Distribution agent randomly recruits some clients to probe suspected peers. Collusion is reported when a peer replies clean content to an illegal download request

Distribution agents randomly recruit clients, called *decoys*, to send illegal download requests to suspected peers. If an illegal request is returned with a clean file chunk, the decoy reports the collusion event. Since the decoy is randomly chosen, there exists a risk that the report is not trustworthy either by error or by cheating. Thus we need a reputation system to screen the peers.

To support the choice of honest decoys, we designed a lightweight reputation system. Consider a P2P network with n paid clients. We use a peer *collusion vector* $\boldsymbol{C} = \{c_i\}$, where $0 \le c_i \le \varphi$ is the collusion rate of peer i. The maximum collusion rate φ is a *collusion threshold,* meaning that any peer exceeding φ is identified as a colluder, Detected colluders are barred from getting new tokens.

When a current token expires, the colluder is labeled as a pirate with denied access to the file. We define a *trust vector* $\boldsymbol{T} = \{t_i\}$, where $t_i = 1 - c_i / \varphi$ for all $1 \le i \le n$. When a decoy i probes a peer j for collusion, it sends j an illegal request and send report r_{ij} to the agent. The condition $r_{ij} = 1$ when j replies with a clean content. The *collusion rate* for peer j is computed by the following expression:

$$c_j = min\{c_j + t_i \times r_{ij},\ \varphi\} \text{ for all } 1 \le i, j \le n. \quad \textbf{(1)}$$

Peer i is identified as a colluder, when its collusion rate exceeds the threshold, i.e. $c_j \ge \varphi$. With this reputation system, a distribution agent weighs each decoy' report against its own trust score to determine the trustworthiness of the reported collusion event. Such a design ensures that a pirate will never be selected as a probing decoy.

Consider a case when the collusion threshold is set with φ=2.5. Consider an honest peer i with an initial collusion rate $c_i = 0$ and thus a complete trust $t_i = 1$ initially. A suspected client j has collusion rate $c_i = 1.6$. We recruit i to probe j, and i reports with $r_{ij} = 1$. We can identify peer j as a colluder since c_j = Min [1.6+1x1, 2.5] = 2.5. This way, only high-reputation clients are hired as probing decoys. Thus more credibility is given to ensure the accuracy of colluder detection.

4. PEER AUTHORIZATION PROTOCOL

In a P2P content distribution network, only the content owner can verify the userID/password pair; peers cannot check each other's identity. Revealing a user's identity to other peers violates his or her privacy. To solve this problem, we developed a PAP protocol. First, we apply IBS to secure file indexing. Then we outline the procedure to generate tokens. Finally, we specify the PAP protocol that authorizes file access to download by peers.

4.1 Secure File Indexing

In a P2P file-sharing network, a file index is used to map a *fileID* to a peer endpoint address. When a peer requests to download a file, it first queries the indices that match a given *fileID*. Then the requester downloads from selected peers pointe4d by the indices. To detect pirates from paid clients, we propose to modify file index to include three interlocking components: an *authorization token,* a *timestamp,* and a *peer signature.*

Each legitimate client has a valid token assigned by its bootstrap agent. The timestamp indicates the time when token expires. Thus the peer needs to refresh the token periodically. This short-lived token is designed for protecting copyright against colluders. The cost at each distribution agent to refresh the client tokens is rather limited, as shown via experiments. The peer signature is signed with the private key generated by PKG. This signature proves the authenticity of a peer.

Download requests make explicit references to file indices. The combined effects of the three extra fields ensure that all references to the file indices are secured. Peers identify the pirates by checking the validity of the token and the signature in a file index. These features secure the P2P network operations to safeguard the sharing of clean contents among the paid clients.

4.2 File-level Token Generation

First, both the transaction server and the PKG are fully trusted. Their public keys are known to all peers. The PAP protocol consists two integral parts: token generation and authorization verification. When a peer joins the P2P network, it first sends authorization request to the bootstrap agent. All messages between a peer and its bootstrap agent are encrypted using the session key assigned by the transaction server at purchase time.

The authorization token is generated by Algorithm 1 specified below. A token is a digital signature of a 3-tuple: {*peer endpoint, file ID, timestamp*} signed by the private key of the content owner. Since bootstrap agent has a copy of the digital receipt sent by transaction server, verifying the receipt is thus done locally. The *Decript*(*Receipt*) function decrypts the digital receipt to identify the file λ. The *Observe*(*requestor*) returns with the endpoint address p. The OwnerSign(λ,p,t_s) function returns with a token.

Upon receiving a private key, the bootstrap agent digitally signs the *fileID,* endpoint address, and timestamp to create the token. The reply message contains a 4-tuple : {*endpoint address, peer private key, timestamp, token*}. The reply message from bootstrap agent is encrypted using the assigned session key.

Algorithm 1: Token Generation
Input: Digital Receipt
Output: Encrypted authorization token ***T***
Procedures :

```
01: if Receipt is invalid ,
02:    deny the request;
03: else
04:    λ= Decrypt(Receipt);
          // λ is file identifier decrypted from receipt //
05:    p = Observe(requestor);
          // p is endpoint address as peer identity//
06:    k = PrivateKeyRequest (p);
          // Request a private key for user at p //
07:    Token T = OwnerSign(f, p, ts)
          // Sign the token T to access file f //
08:    Reply = { k, p, ts, T }
          // Reply with key, endpoint address,
             timestamp, and the token //
09:    SendtoRequestor { Encrypt(Reply) }
          // Encrypt reply with the session key //
10: end if
```

The cost at each distribution agent to refresh the tokens is rather limited. In our experiments, there are 10 distribution agents to serve 1,000 clients/colluders. Each token refresh requires transmitting at most 2 KB of data and each peer is required to refresh its token in every 10 minutes. Per each agent, there are 1000/10=100 peers refreshing tokens in 10 minutes, Hence, we need to transmit only 100 x 2KB = 200 KB to refresh the tokens in every 10 minutes. Considering a standard broadband link capacity of 1.5 Mbps bandwidth, such a low refreshing overhead is negligible.

4.3 The Peer Authorization Protocol

The PAP protocol is formally specified below. A client must verify the download privilege of a requesting peer before clean file chunks are shared with the requestor. If the requestor fails to present proper credentials, then the

client must send poisoned chunks. This procedure is illustrated in Fig. 5.

In PAP, a download request consists of the following elements: token T, file index ϕ, timestamp t_s and the peer signature S. If any of the fields are missing then the download procedure fails trivially. A client receiving a download request must verify both token T and peer signature S. Two pieces of critical information are needed: public key K of PKG and peer endpoint address p.

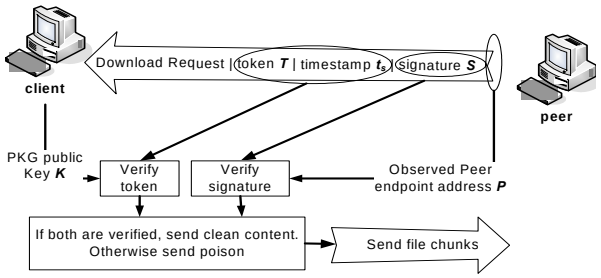


Figure 6 The PAP enables instant detection of a pirate upon submitting an illegal download request.

Algorithm 2 verifies both token T and signature S. File index $\phi(\lambda, p)$ contains the peer endpoint address p and the *fileID* λ. Token T also contains the file index information and t_s indicating the expiration time of the token. The *Parse*(*input*) extracts timestamp t_s, token T, signature S, and index ϕ from an download request. The function *Match* (T, t_s, K) checks the token T against public key K. Similarly, *Match*(S, p) grants access if S matches with p.

Algorithm 2: Peer Authorization Protocol
Input: T = token, t_s = timestamp, S = peer signature, and $\phi(\lambda, p)$ = file index for file λ at *endpoint* p
Output: Peer authorization status
True: authorization granted
False: authorization denied
Procedures :

```
01: Parse (input) = { T, t_s, S, φ(λ, p) }
    // Check all credentials from a input request //
02: p = Observe(requestor);
        // detect peer endpoint address p //
03: if { Match (S, p) fails },
        //Fake endpoint address p detected //
        return false;
04: endif
05: if { Match(T, t_s, K) fails },
        return false;
        // Invalid or expired token detected //
06: endif
07: return true;
```

When a client downloads a file, it needs to authorize the peer to share the file. Otherwise, downloading from a pirate may be poisoned, as shown in Fig.4. When responding queries from honest peers, a client adopts a slightly reduced version of Algorithm 2: Because the inquiry is sent directly to endpoint p, the Observe() procedure is no longer required.

4.4 Adversary and Security Analysis

In contrast to a security-via-obscurity scheme, the PAP protocol is designed to be completely open. We provide an adversary analysis for security assurance of the proposed copyright-protected P2P networks. These assurances ensure that our PAP protocol is secured from common attacks as explained below.

A. Peer endpoint address is forgery proof:

Collusive piracy is achievable, only if the pirate manages to communicate with other peers. IP spoofing can change pirate's endpoint address, resulting in pirate not to receive any response. Therefore, spoofing endpoint address during download is useless to a pirate. A pirate can intercept the token sent to a client, and masquerade its own endpoint address to match with the token. However, using the Observe() subroutine illustrated in Fig.2, other clients will notice the masqueraded peer identity and fail its endpoint verification.

***B. Authorization tokens cannot be shared by peers*:**

A token is generated after the verification of a digital receipt. It establishes the authorization of a client to download and distribute content. It is designed to be a digital signature of a 3-tuple: {*fileID, endpoint address, timestamp*}. Multiple peers cannot share this 3-tuple because each peer has a different endpoint address. Sharing the same token on different endpoint addresses will result in signature mismatch. This also applies to the situation where a pirate steals token from other clients.

***C. Pirates cannot poison legitimate clients*:**

Our system modifies file index format to include tokens and signatures. When downloading from other peers, a client checks the file index for valid signatures. It only downloads file chunks from other legitimate clients that publish some valid file indices. Therefore, even if a pirate attempts to poison other peers, no legitimate client will use it as a download source.

D. Stolen private keys are useless to pirates:

A pirate may hack into a peer's host to obtain its private keys. A colluder may even share these secrets with a pirate. However, sharing or stealing private keys does not help the pirate at all, because of the use IBS endpoint address as public key. Since other clients use Observe() subroutine to obtain peer endpoint address, stolen private keys can never be useful.

5. Protection Performance Analysis

In this section, we analyze the performance of the P2P copyright protection scheme. First, we give the condition to secure the file index. Then we calculate the *poisoning rate* δ of receiving poisoned chunk in response to a pirate's download request. Finally, we estimate the average file *download times T by* legitimate clients and by detected pirates for comparison. The *protection success rate* β measures the percentage of pirates that fail to download the requested file within a given tolerance threshold.

5.1 Secure File Indices

In current P2P networks, a *file index* $\phi(\lambda, p)$ associates a file identifier λ with a peer endpoint address p. In PAP, we replace this index format with a four-tuple:

$$\Phi = \{\phi(\lambda, p), T, t_s, S\} \quad (2)$$

This security-enhanced index format cannot be forged. Both T and S are collision-free signatures. A pirate cannot create its own token or signature via brutal force attack. Therefore, a pirate cannot create index by itself. With Algorithm 2, attempt to modify any single element of $\boldsymbol{\Phi}$ will fail in token or signature verification or both. Therefore, the enhanced index $\boldsymbol{\Phi}$ is secured.

Based on above discussion and section 4.4, there exits a one-to-one mapping of $\boldsymbol{\Phi}$ and client digital receipt. This forgery proof mapping is the foundation of our PAP protocol because it ensures distributed pirate detection at every client. Securing the digital receipt belongs to the realm of general network security, which is beyond the scope of this paper.

The reason for using IBS instead of widely adopted PKI service is due to concern of overhead. In a P2P network with n peers, each peer may need to contact all n-1 peers. If we use PKI service for signature verification, the total CA communication overhead is $O(n^2)$. With an IBS system, this overhead is reduced to $O(n)$, because a peer needs to contact the PKG only once.

5.2 Chunk Poisoning Rate

Our system has an integral function to randomly detect colluders. However, such effect could never be perfect. It is always possible that some colluders will evade the detection. Therefore, these undetected colluders become the real source of copyright violations. Let *collusion rate* ε be the percentage of paid clients acting as undetected colluders. The pirate receives clean content from undetected colluders.

Under a randomized policy, the piracy rate r is the percent of pirates among all peers in the content delivery network. We define *chunk-poisoning rate* δ as the probability of a pirate to receive a poisoned chunk. The following two theorems are obtained.

Theorem 1:

In BitTorrent-like network, the chunk poisoning rate $\boldsymbol{\delta}$ is expressed by :

$$\delta = (1- r)\,(1 - \varepsilon) \qquad \textbf{(3)}$$

For eMule and Gnutella, the chunk poisoning rate δ is expressed by:

$$\delta = 1 - \varepsilon \qquad \textbf{(4)}$$

Proof: In BitTorrent, only a honest client or a distribution agent can poison a pirate. There is no propagation of poisoned chunks among the pirates. The term (1-r) represents the percentage of non-pirates among all peers. Among these peers, (1-ε) is the percent of non-colluding clients. Therefore, $\boldsymbol{\delta}$ is just the product of the two terms.

A pirate cannot identify poisoned file chunks in eMule and Gnutella. The pirate stores undetected poisoned chunks in its local cache and unknowingly shares them with other pirates. We can express poisoning rate by $\boldsymbol{\delta}$ = Probability {poisoned by a client or a agent} + Probability {poisoned by a pirate} = (1-r)(1-ε) + rδ = 1-ε. ***Q.E.D.***

Figure 7 plots the poisoning rate δ in Eq.(1) as a function of r and ε. This is a concave upward surface in the 3-D space. The peak of the protection surface is at the point when we have a fully trusted P2P network by which r = 0. The lowest point corresponds to a completely pirated network where law-abiding peer does not exist.

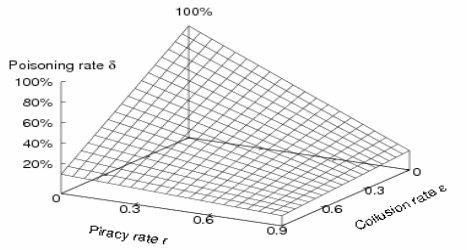


Figure 7 The concave upward surface shows how the poisoning rate in BitTorrent-like networks varies with respect to r and ε.

5.3 Prolonged Download Time by Pirates

In this study, we ignore the accidental corruption of content (*content pollution*) due to transmission error, etc. To capture the inherent poisoning resistance of a P2P network, we define a *piracy penalty* π as the percent of downloaded chunks that are discarded due to poisoning.

Let f be the size of a clean content file and d be the actual file downloaded including clean and poisoned chunks. The ratio f/d represents the percent of downloaded chunks that are clean. Thus, we have:

$$\pi = 1 - f/d \qquad \textbf{(5)}$$

Piracy penalty implies extra workload imposed on the pirates to receive some poisoned chunks. In estimating this piracy penalty, we ignore the effects of network topology, traffic congestions, peer locations, etc. To normalize the results, we consider a *chunk* the smallest unit of a file, whose hash value is used to verify its authenticity and integrity. Hence a chunk refers a *piece* in BitTorrent family and a *chunk* in either eMule or Gnutella families of P2P networks.

We model the piracy penalty on all three P2P content networks: Gnutella, eMule, and BitTorrent. In the eMule network, every 53 chunks form a *part.* Peers compute the hash values of all parts and exchange the part-level *hashsets* inside the P2P network. Thus the part hashsets are also susceptible to content poisoning.

We estimate below the download time of a pirate. This estimation will be verified by simulation experiments. Consider a peer attempting to download a content file of size f. Let b be the average download speed for a peer. A legitimate client does not receive any poisoned chunks. Thus, the download time of a legitimate client is simply calculated by $T_c = f/b$.

By definition of π, 1-π represents the percent of useful download effort. Every time a pirate attempts to download a file, only 1-π portion of the downloaded file is clean and useful. Therefore, to receive the entire file, the pirate must repeat the download attempts $1/(1-\pi)$ times. This leads to the following download time estimated for a pirate T_p :

$$T_p = f/b(1-\pi) \qquad \textbf{(6)}$$

***Theorem 2* :**

The pirate is expected to experience the following *download times* in three existing networks with our proactive copyright protection system:

$$E[T_p] = \begin{cases} f/(b\,\varepsilon^{m}), & \text{Gnutella family} \\ f/(b\,\varepsilon^{54}), & \text{eMule family} \\ f/[b\,(r+\varepsilon-r\varepsilon)], & \text{BitTorrent family} \end{cases} \quad \textbf{(7)}$$

***Proof*:**

In Gnutella, the poisoned file must be discarded and downloaded again, even only one chunk is poisoned. Thus $f/d = (1-\delta)^m$. By Eq.(4), we have $\pi = 1-(1-\delta)^m = 1-[1-(1-\varepsilon)]^m = 1-\varepsilon^m$. In eMule network, a pirate verifies file chunks at the part level. If the part hashset is clean, then poisoning rate $\pi=(1-\delta)^{53}$. In case of a poisoned hashset, we get 100% piracy penalty. Combining these two cases, we have $\pi = \delta+(1-\delta)[1-(1-\delta)^{53}]=1-(1-\delta)^{54} = 1-\varepsilon^{54}$ by substituting δ from Eq.(4).

In BitTorrent, the piracy penalty for the entire file equals the poisoning rate of individual chunk: $\pi = \delta =(1-r)(1-\varepsilon)$ after substituting δ by Eq.(3). Substituting the above results on piracy penalty to Eq.(6), we obtain three expressions in Eq.(7). ***Q.E.D***

We define a *tolerance threshold* θ as the maximum time any pirate can tolerate to download a file. The *protection success rate* β measures the probability that a pirate fails to download the file successfully within the time frame θ. Let $g(T_p)$ be probability density function of the download time by a pirate. Then β is defined as follows:

$$\beta = \Pr ob[T_p > \theta] = \int_{\theta}^{\infty} g(T_p)\,dT_p \quad \textbf{(8)}$$

The above expressions for the poisoning rate δ, piracy penalty π, expected download times T_c and T_p, and protection success rate β are obtained by analytical reasoning. Their accuracy is verified by simulation experiments in Section 6, except in some cases where the pirate download time becomes so large that cannot be simulated in finite time. For simplicity, we assume that pirates adopting a random peer selection policy. In section 6, we will discuss other peer selection policies.

6. SIMULATED P2P EXPERIMENTAL RESULTS

It is very difficult to conduct copyright violation experiments in a real-life P2P network. We have to use simulated P2P experiments to verify the analytical results just presented. We simulate trusted P2P network architectures of three major P2P network families introduced in Table 2.

The trusted P2P features are built into simulators of these three popular P2P networks. The simulation process consists of three stages: First, we measure the chunk poisoning rate δ on simulated networks. Second, we measure the download time and protection success rate of P2P networks simulated. Finally, we compare their differences in defense performance and protection overhead experienced.

6.1 Simulation Setting and Experiments

The simulated P2P network environment is illustrated in Fig.8. The simulator consists of three layers: The lowest layer for data collection and reporting. The middle layer is needed to simulate the behaviors of 4 peer categories: distribution agents, legitimate clients, peer colluders, and illegal pirates. The upper layer simulates the P2P transport.

For eMule simulation, we bypassed the eMule server connections so that the network operates in a completely decentralized mode. Assume that all peers are connected via broadband connections with bandwidth limit of 1.5 Mbps. For simplicity, we ignored network delays and transmission errors. The simulator excludes the communication between distribution agents and the PKG, which are outside of the P2P network. The network topology is also known to all peers.

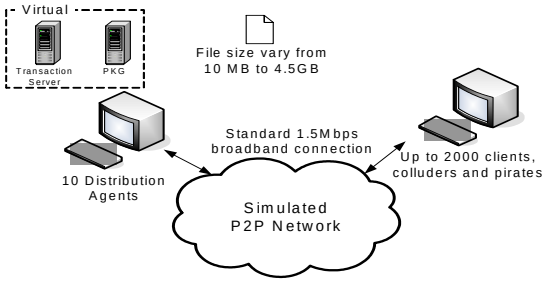


Figure 8 Copyright-protected P2P simulation environment

These assumptions facilitate a fair comparison of the performance of different networks in resisting chunk poisoning. Under these conditions, the simulator measures the shortest possible download time by a pirate or by a client. In reality, the download time by a pirate should be even longer than the measured value due to delays caused by other network factors.

The simulator starts with only the distribution agents having clean file chunks. The file chunks are distributed using a rarest-first policy[20]. We use 10 distribution agents in our simulated networks. By varying the numbers of peers, colluders and pirates, we simulate the P2P network with different level of piracy challenges.

There are many other peer selection policies used by P2P client software. Essentially, these policies favor one group of peers against another. Therefore, we simply lump their effects into different values of peer collusion rate ε or of the piracy rate r.

A P2P network may have millions of peer nodes. Our experiences show that per content file, a peer accesses at most a few hundreds of peer nodes. Since the trusted P2P offers file-level copyright protection, it is sufficient to simulate a P2P overlay with 2,000 of peer nodes. We test the distribution of a 700 MB CD-ROM file. In eMule network, this file is divided into 4,017 chunks.

To explore the limits of a trusted P2P network with the proposed copyright protection features, we also tested small files of sizes 10 MB and 20 MB, and very-large content files of size 4.5 GB of a movie. Based on estimation in Theorem 2, the download times of large content files by pirates may require millions of years, meaning statistically infeasible for a pirate to download a clean file from a trusted P2P network.

6.2 Results on Chunk Poisoning Rate

In all experiments, the chunk poisoning rate δ starts from 100%. This reflects the fact that initially only the distribution agents have some clean file chunks in local caches. Since distribution agents must poison any request from a detected pirate, the pirate will receive only poisoned chunks accordingly.

When some colluders have the chunks to share, the poisoning rate δ is lowered to a stable value after about 2 hours. Figure 9(a) reports the measured poisoning rate δ for the first two hours of experiments on a simulated eMule network. The y-axis shows the average δ value over all 1,000 pirates. The x-axis is time of observation.

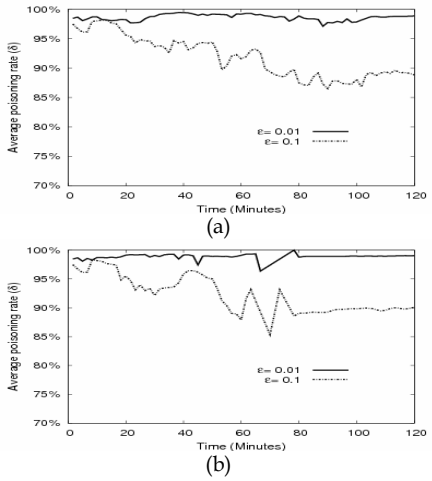


Figure 9 Chunk poisoning rate (δ) in two P2P networks under two collusion rates and a fixed piracy rate r = 50%. (a) eMule network and (b) Gnutella network.

We plot the average δ under two peer collusion rates ε= 1% and 10%. The piracy rate r assumes the value 50%. With low ε= 1%, the poisoning rate is fairly close to the ideal case of δ = 100% of poisoning. With a higher ε = 10%, poisoning rate δ starts at a value close to 1 and fluctuates in a decreasing trend. In about 1.5 hours, δ converges to an estimated value of 0.9. Figure 9(b) plots similar results on the Gnutella network.

Figure 10 plots the effects of changing ε and r on the poisoning rate δ. Figure 10(a) reports results of varying the collusion rate ε with r = 50% and 1,000 pirates out of 2,000 peers. Increasing the collusion rate ε results in a sharp drop of the poisoning rate on both eMule and Gnutella networks.

As the number of colluders increases to 800 (ε = 0.8), the average δ is reduced to only 20%. The decrease of δ is linear with respect to the increase of ε, especially in the case of Gnutella. On the eMule network, the measured poisoning rate δ deviates a little from the theoretical prediction, as dictated by Eq.(2).

The poisoning rate is insensitive to the increase of pirates as shown in Fig.10(b). These results are plotted for the low collusion rate ε = 10% corresponding to 100 colluders out of 1,000 paid peers. The simulation results show that when the number of pirates increases from 200 to 1000, the change of δ is very small.

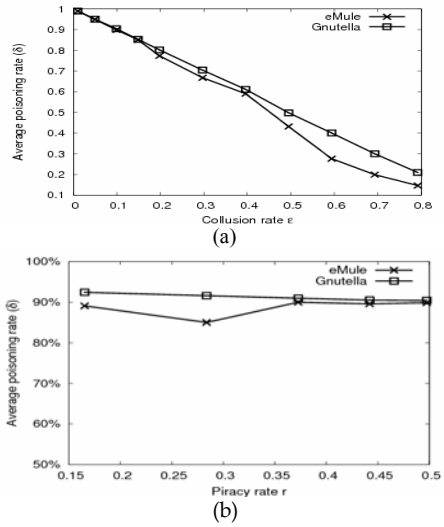


Figure 10 Poisoning rate decreases linearly with increasing peer collusion rate, independent of the piracy rate. (a) Effect of ε *under* r =50%. (b) Effect of r under ε =10%

This result proves the prediction by Theorem 1, namely the poisoning rate δ is irrelevant to the piracy rate r on the eMule network. Although we simulated ε up to 0.8, such extremely high collusion rate is unlikely to occur in real-life P2P applications. We incorporate randomized detection to identify colluders. The detected colluders will not obtain new authorization token, and thus treated as new pirates after the current token expires.

6.3 Download Time by Legitimate Clients

Figure 11 plots the average client download times of a content file with a file size f = 700 MB. Because clients are not poisoned, we measure their download time as a basis for studying the download time penalty of pirates.

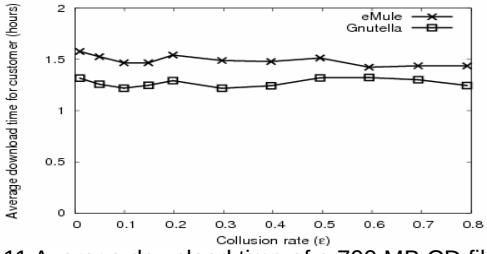


Figure 11 Average download time of a 700 MB CD file by legitimate clients in simulated eMule and Gnutella networks with r = 50% piracy rate.

In both eMule and Gnutella, the average download time for clients remain quite flat around 1.5 hours. This flat download time implies no suffer by legal users under different collusion rates. In Fig.11, a paid client on the Gnutella downloads in 10% shorter time then a client on the eMule network. The shorter download time on Gnutella is attributed to the lower complexity of the file chunking protocol applied.

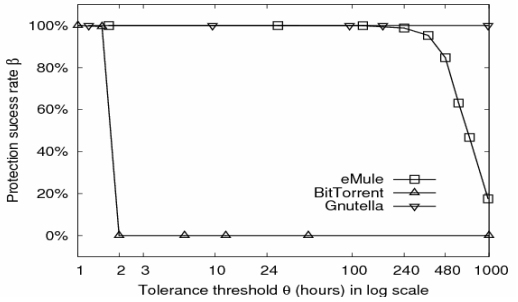
6.4 Pirate Download Time and Success Rate

Now, we report simulation results on the expected download time by pirates. To explore the limit of the trusted P2P system, we have experimented on various file sizes. We define the *protection success rate* β by the failure

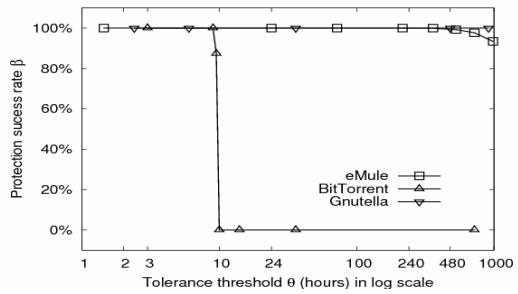
rate of pirates to download the file within a *tolerance threshold* θ of time. The θ is set at 20 days for a 700 MB CD-ROM image file and 30 days for a 4.5GB movie file.

In Fig.12, we simulated all three P2P network families with 100 paid clients, 900 colluders, and 1,000 pirates. This scenario of 90% colluders is very unlikely in a real-life P2P network. We design this experiment to approximate a worst-case scenario that all pirates adopt an aggressive peer selection policy. We measure the percentage of pirates that fail to download a clean copy within the time frame θ as an approximated success rate β. All β curves start with 100% initially. As the threshold increases, we evaluate the success rate in three networks.

Without poisoning, the average download time for a client is 1.5 hours. The Gnutella family, including KaZaA and LimeWire, has a perfect success rate higher than 99.9%. The eMule network has an average 85% after tolerating up to 20 days. These success rates are considered satisfactory due to the fact that most pirates will give up after a few days without success.



(a)



(b)

Figure 12 Protection success rate of a simulated eMule network. Most users have a tolerance threshold far less than 5 days. (a) 700 MB CD-ROM file., (b) 4.5 GB movie file.

The Gnutela family including KaZaA and Lime Wire network has the highest success rate close to 99.9% for both file sizes. With a tolerance threshold of 20 days or 480 hours in Fig.(123), the eMule network has an average of 98% success rate for the 4.5 GB file and 85% success rate for the 700 MB file. These data should be also close to protecting the similar Shareaza networks.

In comparison, an average pirate in BitTorrent network takes about 100 minutes to download the 700 MB file, only marginally longer than a paid client. Hence the success rate drops rapidly before 2 hours. This implies that our system does not protect BitTorrent well due to its strong resistance to content poisoning.

Table 4 reports the *expected download time* T_c by a regular client and T_p expected by a pirate. These numbers are averaged over 1,000 pirates out of 2,000 peers. Some extraordinarily large download times by pirates on eMule or Gnutella networks will exceed hundreds or thousands of years, far beyond what the simulation experiments can do. Their magnitude are calculated using Eq.(4) and marked as greater than 1,000 or 1,000,000 years in Table 3. This implies that pirates on Gnutella networks are impossible to download the file successfully.

Among the three P2P network families in Table 1, we find Gnutella family is best protected by our system. The eMule family is the next and BitTorrent-like networks are most poison resistant. In reality, the tolerance threshold by average pirates is only a few days at most. After that, the pirates may try other means to steal from public networks. That kind of attacks is beyond the scope of our P2P protection scheme.

In general, we find that our system protects eMule and Gnutella families rather satisfactorily, if the tolerance threshold is set to be 20 days for files up to 700 MB. For the 4.5 GB movie file, the tolerance level may have to set as long as 4 months. That implies that all pirates will give up the attempt on P2P networks, if they have wait for so long and still cannot download a clean copy.

TABLE 4 EXPECTED DOWNLOAD TIMES BY PIRATES AND PAID CLIENTS IN THREE SIMULATED P2P NETWORKS

P2P Network	Peer Type	Content File Size	
		700 MB CD File	4.5 GB Movie File
BitTorrent family	Client	94 min	9.2 hours
	Pirate	100 min	9.7 hours
eMule family	Client	1.5 hours	9.5 hours
	Pirate	28 days	4.5 months
Gnutella family	Client	1.5 hours	9.4 hours
	Pirate	> 1,000 years	> 1,000,000 years

6.5 Overheads in Token and Poison Processing

In our experiments, a distribution agent assigns new tokens to about 100 clients/colluders in every 10 minutes. Figure 13 plots the traffic distribution of three message types from a single agent. By Table 4, the regular download time of a 700 MB CD image file to a paid client is about 90 minutes.

We plot the variation of the traffic distribution in 20-minute interval for 2 hours. The white bars for actual content delivery dominates the distribution, consuming almost all 99.9% of maximum link bandwidth of 1.5 Mbps until all clients finished downloading at 90+ minutes. The crossed bars are the total byte count (200 KB estimated in Section 4.2) for token distribution.

Overall, during the content upload period, the token and poisoning overhead is less than 0.1 % (= 0.1 MB/100 MB), which is negligible. The responsibility of content poisoning is distributed to all paid clients and distribution agents. This distribution inevitably costs some upload bandwidth by agents and clients. Compared to clean (un-poisoned) file sharing, the upload bandwidth constitutes the network overhead for distributing poisoned chunks.

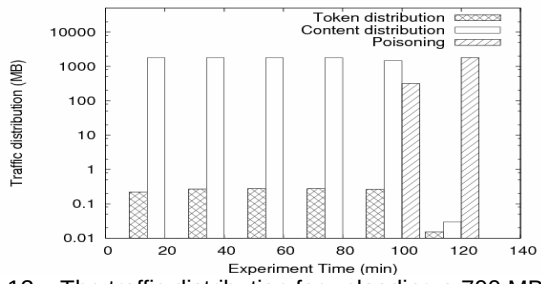
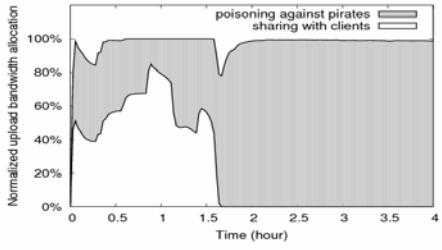
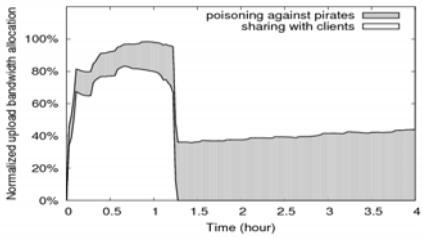


Figure 13 The traffic distribution for uploading a 700 MB CD file along with token and poisoning overheads by a distribution agent to 100 peer nodes

We distinguish in Fig.14 the bandwidth consumed to deliver clean file versus poisoned file chunks. We report the normalized bandwidth allocation for both types of file chunks in eMule network under two peer distributions. The shaded area represents the upload bandwidth of a client allocated to distribute poisoned chunks. The white area is the bandwidth used to distribute the content files to legitimate clients. Initially, since no client has any file chunks to share, the clients receive no request for either.



(a)



(b)

Figure 14 Normalized upload bandwidth needed to distribute clean and poisoned chunks to clients and pirates, respectively, in a simulated eMule network. (a) High piracy rate (50%), (b) Low pirate distribution (10%).

Figure 14(a) shows a high piracy rate of r = 50%, meaning 1000 pirates are present out of 2000 peers in the simulated eMule network. Therefore, almost all upload bandwidth is used to handle the pirate requests after 1.5 hours. Figure 14(b) corresponds to a low piracy rate of 200 pirates out of 2000 peers. Thus lower portion of the bandwidth is allocated to distribute poisoned chunks. The poisoning overhead is dropped to 40% after one hour.

Figure 15 plots the upload bandwidth allocated for content poisoning at a distribution agent. The agents dedicate the upload bandwidth, when client download is in progress. When there are no legitimate requests, distribution agents are fully dedicated to the task of poisoning pirates. This switching of bandwidth allocation enhances a low-cost P2P file distribution network.

For a small content file of 10 MB, the switching of bandwidth usage takes place at 2 minutes; The 50 MB file takes 7 minutes to switch from content delivery to complete poisoning. The 700 MB file takes 90 minutes to switch the bandwidth allocation. The differences in the switch time correspond to the average download time of a legitimate client.

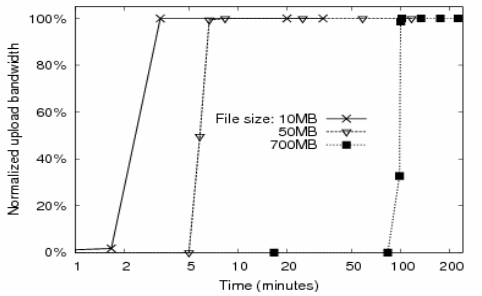


Figure 15 Normalized upload poisoning overhead by distribution agents in delivering files from 10 MB to 700 MB over the ADSL links.

7. Conclusions and Further Research

Through evidences from performance analysis and experimental results, we have demonstrated that the combined use of IBS-based indexing and selective chunk poisoning are indeed effective to detect colluders and pirates and stop them from collusive piracy in major families of P2P networks.

7.1 Major Research Findings

Summarized below are the major research findings and discussions on deployment requirements of our proposed P2P copyright protection system:

A. Stop piracy at the presence of colluders:

With secure file indexing and assistance from a peer reputation system, colluding peers are detectable. The system stops piracy by poisoning pirates with excessively long download overhead at the presence of a large number of colluders.

***B. Pirates detected immediately upon first attempt*:**

With the timestamped authorization token, the PAP protocol enables clients to detect illegal download attempts from a pirate without communicating with a central authority. Our scheme heavily penalizes copyright violators without hurting honest clients.

***C. Selecting high-reputation peers as probing decoys*:**

Using a reputation system, we select trusted clients to act as decoys to probe peers in collusive piracy. This mechanism accurately detects colluders when they send clean file chunks to illegal requests. Reducing number of colluders can improve the protection against pirates, as demonstrated by experiments.

***D.Gnutella-like networks best protected by our scheme*:**

Applying our protection scheme, the Gnutella family, including Gnutella, Ares, KaZaA, LimeWire, Freenet, BareShare, etc., demonstrates the highest penalty on

pirates because poison detection is only possible at the file level. Even a few chunks poisoned, the entire file must be discarded and downloaded repeatedly.

This makes the file download time by pirates intolerably long (1,000 years or longer). Thus, this family of P2P networks are most suitable to apply the proposed copyright protection scheme, yielding almost a perfect protection success rate (> 99.9%) reported in Section 6.4.

E. eMule-like networks get protected satisfactorily :

The eMule-like P2P networks apply weak chunk hashing at the part level. Pirating on eMule networks still experiences very high download overhead in days or months as seen in Table 4. Our copyright-protection system works with 85-98% success rate on eMule family of P2P networks. Our protection scheme will be applicable to eMule, Shareaza, Porpheus, and FastTrack as listed in Table 2.

E. BitTorrent-like networks most resistant to poisoning **:**

Our system is less effective to protect poison-resistant P2P networks like BitTorrent, BNBT, Azureus, etc. Continued research is needed to extend the PAP protocol to overcome this difficulty. On BitTorrent-like networks, if the pirate repeats the download attempts many times, eventually they may succeed in getting the clean file within a practical time frame.

F. Protection scheme performs better for large files:

The proposed system is more effective to protect large files. Our PAP protocol is not tailored to protect short music or document files. A popular song in MP3 format has less than a few MB in size. Content poisoning take less effects on such small files due to single or a few chunks contained in the file.

G. Negligible detection and poisoning overhead:

The proposed PAP protocol detects colluders and pirates and apply chunk poisoning selectively. These extra activities add only limited extra workload or traffic on the network. More importantly, these overheads are distributed among all distribution agents and clients, which further reduces their effects on individual clients.

7.2 Discussions and Suggestions

Our protected content delivery always gives higher priority to satisfy honest clients. Putting poisoning tasks at the lower priority reduces the upload overhead. Our selective chunk poisoning outperforms the un-discriminated poisoning practiced by RIAA or MPAA enforcers. Our protection system is fair to the majority of honest clients who enjoy P2P content delivery services.

Perishable content has its utility and value diminishing quickly after it is published. Perishable contents, such as real-time broadcast of news or sport events, are well protected by our system. A hashing mechanism to detect poisoning cannot be effective, because distributing chunk hashes ahead of content is impossible in real time. Essentially, all P2P networks including BitTorrent fail to resist poisoning in perishable contents.

Existing DRM systems [10] focus on enforcing usage rules of digital content such as copy and replay, while our proposed system provides a protected P2P distribution platform to control the access of copyrighted content files. Combining DRM with P2P copyright protection should be the focus of future research.

Once a pirate discovers how the system works, he can break it; post the hack on the Internet and everyone will be able to bypass the security check. In comparison, our system is completely open. A pirate can steal tokens from clients; it may even obtain the source code of our software. Nonetheless, we enforce copyright protection via distributed content poisoning. Statistically, a pirate cannot download the file successfully in finite time if the P2P network is protected by the PAP protocol.

Two concrete R/D tasks are suggested below for further work to combine all means of copyright protection in P2P content delivery.

H. Prototyping and Benchmark Experiments needed in real-life open P2P networks:

Simulation results reported here can only prove the protection concept, lacking of sustained accuracy. Proactive chunk poisoning can be made selectively to reduce the processing overhead. However, further studies are needed to upgrade the performance of the copyright-protected system in real-life P2P benchmark applications.

I. Integration of trusted P2P system with reputation system and DRM scheme:

File-level reputation system posts a new challenge to work with DRM systems in P2P content delivery. The integration of selective poisoning with reputation system and DRM will widen the CDN application domains. Combining DRM and reputation system to further protect P2P content delivery networks will lead to a total solution of the on-line piracy problem.

Acknowledgments: This work was supported by NSF ITR Grant ACI-0325409 at the University of Southern California. The work was carried out at USC Internet and P2P Computing Laboratory. We appreciate the technical support by GridSec team members.

REFERENCES

[1] N. Anderson, "Peer-to-Peer Poisoners: A Tour of Media-Defender", Ars Technica, Sept.16, 2007.

[2] BitTorrent.org, "BitTorrent Protocol Specification". Url: http://www.bittorrent.org/protocol.html , 2006

[3] S. Androutsellis-Theotokis and D. Spinellis, "A Survey of Peer-to-Peer Content Distribution Technologies", *ACM Computing Surveys*, Vol. 36. 2004, pp. 335--371.

[4] D. Boneh and M. Franklin, "Identity-based encryption from the Weil Pairing," *Advances in Cryptology-Crypto'2001*, Springer-Verlag , 2001, pp. 213--229.

[5] S. Chen and X. D. Zhang," Design and Evaluation of a Scalable and Reliable P2P Assisted Proxy for On-Demand Streaming Media Delivery", *IEEE Transactions on Knowledge and Data Engineering,* May 2006, pp. 669-682.

[6] 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.

[7] 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, 2005, pp. 68–77.

[8] 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.

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

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

[11] J. Frankel and T. Pepper, "The Gnutella Protocol Specification v0.4, revision 1.2", 2000, http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf.

[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. Hofmann and I. Beaumont, *Content Networking, Architecture, Protocols, and Practice,* Kaufmann, S.F. 2005.

[14] Y. Itakura, M. Yokozawa and T. Shinohara, "Model Analysis of Digital Copyright Piracy on P2P Networks," *Proc. of International Symposium on Applications and the Internet Workshops* (SAINT), 2004, pp. 84-89, 26-30 Jan. 2004

[15] T. Iwata, T. Abe, K. Ueda and H. Sunaga, "A DRM system suitable for P2P content delivery and the study on its implementation," *The 9th Asia-Pacific Conference on Communications* (APCC), 2003, vol.2, pp. 806-811 Sept. 2003

[16] T. Kalker, D. H. J. Epema, P. H. Hartel, R. L. Lagendijk, and M. Van Steen, "Music2share - Copyright-Compliant Music Sharing in P2P Systems," *Proc. of the IEEE*, Vol. 92, 2004, pp. 961-970.

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

[18] Y. Kulbak and D. Bickson, "The eMule Protocol Specification," *Hebrew University* TR-2005-03, Jan. 2005.

[19] S. H. Kwok, "Watermark-based Copyright Protection System Security" *Comm. of the ACM,* Oct. 2003, pp. 98-101.

[20] A. Legout, G. Urvoy-Keller, and P. Michiardi, "Rarest First and Choke Algorithms are Enough", *Proc. of ACM SIGCOMM on Internet Measurement,* Rio de Janeriro, Brazil, 2006, pp. 203 - 216.

[21] E. Luoma and H. Vahtera, "Current and Emerging Requirements for Digital Rights Management Systems through Examination of Business Networks," *Proceedings of the 37th Annual Hawaii Int'l Conf. on System Sciences* 2004.

[22] D. P. Majoras, O. Swindle, T. B. Leary, and J. Harbour, "Peer-to-Peer File-Sharing Technology: Consumer Protection and Competition Issues", *Federal Trade Comm. Report*, June 2005

[23] N. Mook, "P2P Flooder Overpeer Cease Operation", Beta News, Dec.10, 2005, http://www.betanews.com/article/P2P_Flooder_Overpeer_Ceases_Operation/1134249644

[24] N. Mook, " P2P Future Darkens as eDonkey Closes", Beta News, Sep.28, 2005, http://www.betanews.com/article/P2P_Future_Darkens_as_eDonkey_Closes/1127953242

[25] G. Pallis and A. Vakali, "Insight and Perspectives for Content Delivery Networks," *Comm. of The ACM,* Jan. 2006, pp.101-106

[26] P. Rodriguez et al, "On the Feasibility of Commercial Legal P2P Content Distribution" *SIGCOMM Comput. Comm.*. Rev.36, 1 (Jan. 2006), pp.75-78.

[27] S. Saroiu, et al., "An analysis of Internet Content Delivery Systems," *SIGOPS Operating System Review*, pp. 315–327, 2002.

[28] M. Srivatsa and L. Liu, "Vulnerabilities and Security Threats in Structured Overlay Networks: A Quantitative Analysis," *20th Annual Computer Security Applications Conference*, 2004.

[29] J. Sung, J. Jeong and K. Yoon, "DRM Enabled P2P Architecture", *Proc. of the 8th Int'l Conf. on Advanced Comm. Technology* (ICACT 2006), Vol.1, Pages: 487- 490

[30] 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, 2005, pp. 138–143.

[31] C. Wang, B.A. Alqaralleh, B. B. Zhou, F. Brites, and A.Y. Zomaya, "Self-Organizing Content Distribution in a Data Indexed DHT Network", *Proc. of the Sixth IEEE Int'l Conf. on Peer-To-Peer Computing,* Washington, DC, 2006, pp. 241-248.

[32] 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.

[33] M. Yurkewych, B. N. Levine, and A. L. Rosenberg, "On the Cost-ineffectiveness of Redundancy in Commercial P2P Computing," *Proc. of 12th ACM Conf. on Computer and Communications Security*. Alexandria, VA, 2005.

[34] R. Zhou and K. Hwang, "GossipTrust for Fast Reputation Aggegation in P2P Networks", *IEEE Trans. on Knowledge and Data Engineering* (TKDE), accepted to appear 2008

Xiaosong Lou received the B.S. degree from Shanghai Jiaotong University in 1994. In 2005, he received the M.S. degree in Computer Engineering from the University of Southern California. Currently, he is persuing the Ph.D. degree at USC Computer Engineering Program. His search interest covers peer-to-peer and Grid computing, poisoning detection, distributed content delivery.

Kai Hwang is a Professor of Electrical Engineering and Computer Science at University of Southern California (USC). He received the Ph.D. from the University of California, Berkeley in 1972. An IEEE Fellow, he specializes in computer architecture, parallel processing, Internet security, and distributed computing. He has published 7 books and over 200 scientific papers in these areas. He has produced 20 Ph.D. students at Purdue University and USC over the years.

Dr. Hwang is the founding editor of the *Journal of Parallel and Distributed Computing* and an associate editor of the *IEEE Transactions on Parallel and Distributed Systems.* He has led various research groups at Purdue and USC. He has chaired numerous ACM/IEEE International Conferences and presented over two dozens of keynote addresses in various Conferences. He has lectured worldwide and performed advisory work for IBM Fishkill, Intel Scalable System Division, MIT Lincoln Lab., JPL at Caltech, ETL in Japan, Academia Sinica in China, GMD in Germany, and INRIA in France. For details, contact him via Email: kaihwang@usc.edu or visit his personal web site: http://GridSec.usc.edu/Hwang.html.