Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2014, Article ID 792407, 18 pages
http://dx.doi.org/10.1155/2014/792407

Research Article

Application of Fault-Tolerant Mechanism to Reduce Pollution Attacks in Peer-to-Peer Networks

Chien-Sheng Chen,[1] Ting-Yuan Yeh,[2] Chin-Tan Lee,[3] and Chyuan-Der Lu[4]

[1] *Department of Information Management, Tainan University of Technology, Tainan 71002, Taiwan*
[2] *Department of Engineering Science, National Cheng Kung University, Tainan 70000, Taiwan*
[3] *Department of Electronic Engineering, National Quemoy University, Quemoy 89250, Taiwan*
[4] *Department of Finance, Tainan University of Technology, Tainan 71002, Taiwan*

Correspondence should be addressed to Chien-Sheng Chen; t00243@mail.tut.edu.tw

Received 17 October 2013; Revised 3 April 2014; Accepted 7 April 2014; Published 7 July 2014

Academic Editor: Yang Xiao

File pollution is a recent security threat to peer-to-peer (P2P) file sharing systems. By disseminating numerous polluted files with mismatched or partially tampered contents in the P2P system, the attacker causes users to download unexpected files. This attack is aimed at frustrating users and making them abandon the system. Present researches on combating file pollution have mostly focused on pollution modeling or evaluating the extent of pollution. Only a few researches have proposed effective methods to eliminate pollution attacks, and they are primarily based on reputation systems and blacklisting mechanisms. However, these methods require exchange of significant feedback among the peers in order to identify the malicious peers or polluted files in the system. In this paper, we describe the application of fault-tolerant mechanism used in the redundant arrays of independent disks system to suppress file pollution attacks based on the concept that P2P file sharing systems currently have global file storage systems. We have extended the previously developed Fluid Model to analyze and evaluate the proposed antipollution mechanism. The model accuracy has been demonstrated by performing several simulation experiments; the proposed mechanism could effectively suppress the pollution and successfully decrease the polluted-time exposure of a P2P file sharing system by approximately 40~60%.

1. Introduction

A file sharing system is an important application in peer-to-peer (P2P) networks; it enables files to be shared with high efficiency by allowing parallel downloads of the duplicated file pieces that are distributed across several peers. The robustness of P2P networks is dependent on the total system capacity; as the total system capacity increases, the number of duplication times of resources in more peers also increases, thereby leading to more fault-tolerant systems. Since the release of the Napster system in 1999, P2P file sharing systems such as Kazaa and BitTorrent have become some of the most popular systems on the Internet. The shared contents encompass a wide variety of MP3 songs, movie films, software packages, and games, and they have motivated numerous network citizens (Netizens) to install P2P file sharing packages because of the speed of their sharing and diversity. For example, in the case of Kazaa, which is one of the most popular P2P sharing software packages, Good and Krekelberg [1] pointed out that there were more than 3 million users in 2003 with the shared file size amounting to almost 5000 TB. Today, P2P file sharing flow has become the most important source of network traffic on the Internet [2].

However, among the many people who join the P2P file sharing system, not all of them have an understanding or appreciation of the associated security risks. This makes the P2P file sharing systems easy targets for malicious attackers. Many types of security attacks have already been identified; these include churn attacks, poisoning and pollution, Sybil attacks, worms, malware, and cheating in P2P massively multiplayer online games (MMOGs) [3]. Pollution attack is a major recent security threat that was first proposed by Liang et al. [4] at the INFOCOM in 2005. Pollution attacks occur primarily due to counterattacks against the

illegality of fast and quantitative sharing and dissemination of copyrighted materials [5]. Many musical, movie, software, and publishing industries have suffered significant revenue impacts due to the popularity of P2P file sharing software. Hence, the impacted companies have taken a series of countermeasures [4] including prosecuting instances of copyright infringements by companies and individuals and performing file pollution attacks that significantly reduce the file usability by destroying or tampering file titles, metadata, or contents, thereby making users frequently download unexpected files and, hence, be frustrated and quit the system.

The decentralized management architecture adopted for the sake of scalability and autonomy has made the P2P file sharing system vulnerable to security threats. Moreover, the high efficiency of the file sharing feature leads to pollution attacks [6, 7] and the fast disseminating of other malware and viruses [8] across the networks. In 2005, [4] showed that 50%~80% of the shared files in Kazaa had already been polluted; in 2006, other research [9] demonstrated that roughly 50% of the popular files of other well-known file sharing software, eDonkey, had been polluted.

It is anticipated that the researches and developments of legal downloading mechanisms [10] may finally cause copyright owners to stop the file pollution attack measures taken against the P2P file sharing systems in the name of copyright protection. However, file pollution may continue to be a popular method taken by other malicious attackers to threaten the security of P2P file sharing systems. Hence, in this paper, we focus on the suppression and avoidance of the dissemination of polluted files in P2P file sharing systems. Most current methods have adopted the reputation system [6, 11, 12] or blacklisting mechanism [13] to achieve such goals, but they usually require user feedback information to identify the malicious peers in a system or polluted files. Hence, they have the following issues.

(1) A user has to first occupy the network resource and download the files totally or partially before knowing the contents that have been polluted.

(2) A user, after having downloaded some files, will not typically check immediately whether the files are polluted or not; this allows the polluted files in the sharing folders to be shared and disseminated further.

(3) Feedback information may easily suffer from tampering or forging and thereby cause many additional security issues.

Therefore, in this paper, we have attempted to develop an effective antipollution mechanism without using user feedback. Our method has been developed on the basis of the idea that a file in a P2P sharing system can be transferred after dividing it into segments and then reassembling into a complete file when all the segments are available. We have used a fault-tolerant mechanism adapted from the EVENODD coding scheme [14] that is used in the disk storage technology. This mechanism tolerates disk failures in the redundant array of independent disks (RAID) system effectively and efficiently—each polluted segment is considered to be a failed disk, and with proper rebuilding, a tampered file can be recovered correctly. The degree of tolerance depends on the number of segments into which a file is broken, and a balance between the number of segments and file reconstruction efficiency can be reached after some experiments. Hereafter, the EVENODD coding scheme will be abbreviated as EVENODD, and its operating principle and the reasons for using it will be described in the following sections. This proposed mechanism is capable of avoiding and eliminating the polluted file segments without using user feedback information or changing the P2P network structure. The Fluid Model (FM) [15] has been extended in this paper to construct a model that is able to analyze and evaluate the proposed antipollution mechanism. Moreover, by using simulation experiments, we have demonstrated the accuracy of the proposed model. We have shown that the proposed mechanism can effectively avoid pollution and decrease the polluted-time exposures of P2P file sharing systems by 40~60%.

The rest of this paper is organized as follows. Section 2 discusses the background and related researches that have already been conducted to counter the P2P file pollution. EVENODD is introduced in Section 3. Section 4 describes the system architecture and operations of the proposed antipollution mechanism. The deduction of the mathematical system model is presented in Section 5. Section 6 describes the evaluation results. Section 7 presents equations to amend the system model. Finally, Section 8 summarizes this study.

2. Background and Related Work

The contents that are shared, legally or illegally, on a P2P network are mostly music, movies, documents, and software packages; each of such contents has a ⟨Title⟩, and further, because of the different content attributes, a title has many different ⟨Versions⟩. For instance, for a piano etude, there may be several versions corresponding to different performers, while for an MP3 song, there may be different versions comprising different encoding rates. When these versions are disseminated on a P2P network, the file owned by the peers that acquire that particular version is called the "Copy of this Version."

A user who wishes to download the Copy of a Title from the P2P file sharing system first issues an enquiry to the system. For example, a user who is interested in a song with the ⟨Title⟩ "Hey Ya" makes an enquiry to the system using the string "Hey Ya." The system then responds to the user with all the versions available in the P2P file sharing system and the number of quantities and locations of the copies of each version. The user chooses to download a suitable version according to his preference. The file is downloaded and saved in the user's sharing file folder to be downloaded by other peers in the system. In this manner, the file is disseminated very quickly in the network.

2.1. File Pollution Categories. To initiate a pollution attack, a pollution attacker first produces a polluted version by adopting polluting techniques such as quality reductions or content alterations of the target file. The polluter subsequently

injects this modified content into the system on his peer or on several peers under his control and, by using the title of this polluted version, makes an announcement to the system that is visible to other peers and downloaded as per the requirement. As a result, while searching for a certain title, a user may be confronted with a polluted file and an authentic file. Since it is not possible to verify in advance whether the content has been polluted or not, an unaware user may choose to download the polluted version. Moreover, the user usually may not check the downloaded file that is saved in the sharing folder immediately, which causes the polluted file to be disseminated faster and further in the P2P file sharing system.

In the strategy adopted by malicious attackers, the pollution on a file can be categorized into three types [4, 9].

(1) Content Pollution. Content-modified decoys are created for users to download, which then disseminate in the system. A decoy is created by adding certain noises that cannot be decoded into meaningful content in a target file or by changing the content into that of a different file but retaining the metadata of the original. Since it is difficult to determine whether the file content has been polluted before downloading, this method represents a very simple and effective polluting approach. The suppression and elimination (removal) of these types of pollution are the primary goal of this paper.

(2) Metadata Pollution. In this type of pollution attack, metadata such as the file name and type of a targeted file are replaced with those of a different unrelated file. This causes a user to choose and download a file containing content that does not totally conform to expectations.

(3) Index Poisoning. In a P2P network framework that possesses the capabilities of file searching, for example, a P2P file sharing network based on distributed hash tables, a pollution attacker causes a nonexistent file to be shared on the network by uploading bogus file records such as the IP address of the file sharing person, port number, and file hash value to the supernodes that are responsible for maintaining these records in the network or by modifying the file records.

2.2. File Pollution Studies and Counteractions. The measurement study conducted by [4] focused specifically on Kazaa, the most popular file sharing system, and developed a crawler that could quickly retrieve the supernodes in the Kazaa network; subsequently, from the raw data on popular music collected by the crawler, statistics and observation studies were conducted on the file pollution status. Their results showed that the pollution was very serious; in the Kazaa file sharing system, more than 50% of the popular copies of musical songs were polluted. Subsequently, many researchers have dedicated themselves to conduct measurement studies to examine the extent of the impact of pollution attacks on the P2P network. For example, Liang et al. [9] also developed an effective methodology to estimate the amounts of index poisoning and pollution in the file sharing system; they used data-harvesting platforms on FastTrack and Overnet to collect the required data. Christin et al. [5] provided a measurement-based analysis to inspect the impacts on content availability due to pollution and poisoning in the P2P network. Dhungel et al. [16] conducted experimental measurements to analyze and study the pollution in P2P live video streaming systems.

Other researchers have constructed analysis models to facilitate the analysis and evaluation of the impacts of pollution on P2P networks. Dumitriu et al. [17] performed analytical modeling against file-targeted attacks and network-targeted attacks in order to investigate the resilience demonstrated by a P2P file sharing system when fighting against denial-of-service (DoS) attacks. The study of Shi et al. [18] presumed that the file-targeted attacks and network-targeted attacks have close and inseparable relationships and, in this manner, proposed a unified model for these two attacks. Thommes and Coates [19] inspected the viruses in P2P networks in addition to the polluting behavior and adopted an epidemiological approach to explore a dynamic model capable of describing viruses and pollution. Similarly, from the concept of an epidemic model, Gu et al. [20] proposed a model of reputation-based approaches, which could simulate and evaluate pollution, in order to study the dissemination of a shared file in P2P networks. Lee et al. [21] investigated user behaviors by using questionnaires and then employed the results as a basis to explore an analytical model in order to study the dynamic properties of pollution in P2P networks. In the study of Yang et al. [22], a modeling framework was constructed to analyze and study the pollution in P2P live video streaming systems. Kumar et al. [15] developed the FM for pollution proliferation in a P2P file sharing system; in this paper, we have extended a small part of the FM to construct a model that is able to analyze and evaluate the proposed antipollution mechanism.

Thus far, few researchers have focused on developing pollution countering (measures), and most of them have adopted a reputation system or a blacklisting mechanism. Credence [12] is a well-known reputation system for fighting viruses in P2P file sharing systems. By employing a simple network-wide voting scheme, Credence enables a user to conduct positive and negative appraisals on object contributions in P2P file sharing systems; moreover, it also allows a client to perform weighting on his vote based on the statistical correlations between himself and his peers. In Dong et al. [23] and Dong et al. [11], a P2P antipollution file sharing system was constructed based on object reputations, but employing different building processes. They proposed a pollution propagation model that first considered file objects, then weighted votes by calculating the vector space similarity, processed the data sparsity problem by using the Horting graph-theoretic approach, and finally adopted a self-adaptive reputation threshold to judge the file authenticity. On the other hand, Costa et al. [6] proposed a P2P antipollution file sharing system, named Scrubber, that was based on peer reputations. In comparison to Credence, Scrubber is a distributed and decentralized reputation system inside which peers mutually designate the reputations to verify and differentiate malicious users who disseminate polluted contents on the network. In addition to the imposition of severe and speedy punishments on the content polluters, Scrubber

also includes a rewarding mechanism to recover a peer's reputation. However, the above-mentioned proposals cannot rapidly react to the dissemination of polluted contents. Cai et al. [24] proposed a holistic mechanism to defend against pollution attack. Besides using a general reputation model, the mechanism gathers extra inherent file-related information for peers to identify the potential pollution without completely downloading the requested content. Barcellos et al. [25] developed a reputation-based pollution control strategy in which the dissemination rate of a content is limited according to the reputation of the version. In this way, the polluted content is eliminated before a peer vote on the version negatively. Shin and Reeves [26] presented an antipollution scheme called winnowing. It reduces index pollution by publish message verification and user feedback mediation in DHT-based P2P system.

Liang et al. have proposed certain countermeasures based on a blacklisting mechanism after conducting many measurement studies on P2P network pollution [13]. They had developed the abovementioned crawler that crawls an entire P2P network and collects the metadata of all the shared files for offline analysis. In their automated version-checking procedure, a set of nodes of very probable polluters are verified and added to a blacklist. Further, another study had focused on countering index poisoning attacks in P2P systems; a similar method was followed in [13] via a step named harvesting. In harvesting, massive information relevant to the versions in a P2P system is collected, and the poisoned, polluted, and clean versions are verified; bad sources are then recognized and added to a blacklist. It is possible to suppress the pollution spreading in P2P file sharing systems without downloading the data shared from the blacklisted nodes. However, in blacklisting mechanisms, massive amounts of data have to be collected on a P2P network, and it is not affordable to normal individual users. Moreover, since the peers in a P2P network join and leave the system frequently and unexpectedly, the data search procedures have to be conducted often to prevent the collected data from becoming out of date, and this significantly increases the network overheads.

3. EVENODD Coding Scheme

Reed-Solomon (RS) codes [27] and EVENODD are two representative forward error correction (FEC) codes widely used in telecommunication and information theory, such as data transmission and storage systems. In the RS codes, an update to a single information bit requires an update in all the parity symbols and affects numerous bits in each symbol. As a result, the RS code update operations incur large computation overheads. Unlike RS codes, EVENODD requires only cyclic shifts and XOR operations, which therefore achieves optimal redundancy with considerably lower computational complexity. Furthermore, EVENODD can tolerate up to two errors or erasures of bits (disks) and is more practical than RS codes.

The development of coding schemes remains an active research area. Since the EVENODD was proposed, several schemes have been proposed, such as the remote desktop protocol (RDP) scheme [28], X-code [29], B-code [30], and generalized EVENODD code [31]. These schemes are more efficient than EVENODD, and some of them can tolerate more than even two erasures. However, most of these schemes are variations or extensions of EVENODD, and hence, they are more difficult to implement due to the high complexity for achieving high performances. In this study, we aim to find a fault-tolerant mechanism to reduce the pollution attacks in P2P networks; developing or looking for high-performance coding schemes is beyond the scope of this study. EVENODD is both efficient and simple and hence is suitable for the purpose of this study. In this section, we will briefly introduce the EVENODD encoding scheme. For more details, please refer to [14, 32, 33].

3.1. Encoding Procedure. In a RAID, in order to tolerate the simultaneous damages to two disks, EVENODD requires $(m + 2)$ disks to spread its encoding of stored data; here m must be a prime number. The original data is split into blocks and, after proper encoding, distributed as uniformly as possible into m disks. Each disk is split into $m-1$ blocks, and hence, the RAID system logically forms a $(m - 1) \times (m + 2)$ matrix A. The redundant data generated from the original data for crash recovery are saved in the two extra disks m and $m + 1$. The parity data calculated from the horizontal blocks of disk 0-disk $m-1$ is saved on disk m, while the parity data calculated from the diagonal blocks disk 0-disk $m - 1$ is saved on disk $m + 1$. In array A, $A[i, j]$ represents the block of the ith row and jth column (i.e., the ith block of the jth disk), where $0 \le i \le (m - 2)$ and $0 \le j \le (m + 1)$. Equations (1)–(3) are the formulae for calculating the parity data. For each i, $0 \le i \le (m - 2)$, the parity data saved in disk m can be calculated from

$$A[i, m] = \bigoplus_{j=0}^{m-1} A[i, j]. \tag{1}$$

Here $\bigoplus$ represents an XOR operation. Equation (1) performs an XOR operation on the horizontal block of the original data disk array. The calculation of the parity data saved in disk $m + 1$ uses (2) and (3) as follows:

$$S = \bigoplus_{j=1}^{m-1} A[m - 1 - j, j], \tag{2}$$

$$A[i, m + 1] = S \oplus \left(\bigoplus_{j=0}^{m-1} A[\langle i - j \rangle_m, j] \right). \tag{3}$$

Before calculating the S value, the 0th column of the original data array is removed to make it a square array. XOR operations are then performed on the data from the lower-left to upper-right diagonal blocks in the square array to obtain the S values; subsequently, all the blocks of the original data array are shifted one grid to the left in order to obtain another diagonal line. An XOR operation is then performed on the blocks of this diagonal line to obtain a value, and a single XOR operation is again performed on the just calculated value and S values. The results are saved on

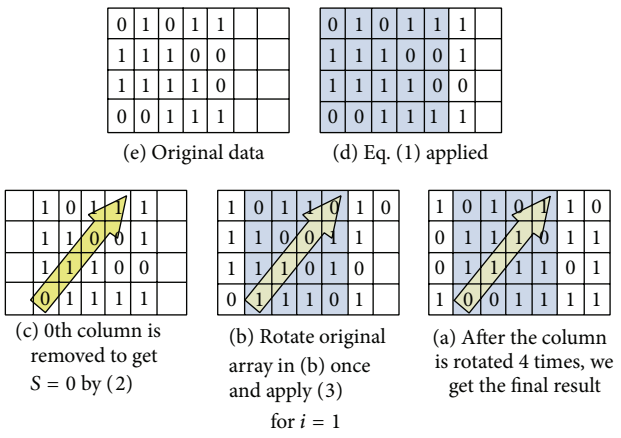

Figure 1: An EVENODD encoding example.

the 0th block of the $m + 1$ disk. Figure 1 shows an example of obtaining the parity data based on the abovementioned calculation process wherein the rotation is shown for easy explanations. For detailed calculation processes, please refer to the example in [14].

3.2. Decoding Procedure. If we consider the situation in which only one disk crashes, the lost data can be easily recovered by using (1), (2), and (3) for reversal computations. When two disks crash simultaneously, 4 situations must be considered. Let us assume that both disks i and j crash, where $0 \le i < j \le (m + 1)$; consider the following situations.

3.2.1. $i = m$, $j = (m+1)$. The damaged ones are the two disks with parity data saved on them. The damaged parity data can obviously be recalculated by using (1), (2), and (3) after the new disks are installed.

3.2.2. $i < m$, $j = m$. One of the two damaged disks retains the original data while the other disk m retains the horizontal parity data. The parity data of disk $m + 1$ and the unaffected diagonal block data in the array are substituted into the following formula to obtain the value of S:

$$S = A\left[\langle i-1\rangle_m, m+1\right] \oplus \left(\bigoplus_{l=0}^{m-1} A\left[\langle i-l-1\rangle_m, l\right]\right). \quad (4)$$

Substitute the calculated value of S into the following formula:

$$A[k,i] = S \oplus A\left[\langle i+k\rangle_m, m+1\right] \oplus \left(\bigoplus_{\substack{l=0 \\ l\neq i}}^{m-1} A\left[\langle k+i-l\rangle_m, l\right]\right). \quad (5)$$

Here, k is every single damaged disk block. These two equations are related to the recovery of the damaged original disk, while the lost horizontal parity data in disk m can be regained by (1).

3.2.3. $i < m$, $j = (m + 1)$. One of the two damaged disks retains the encoded original data while the other disk $m + 1$ retains the diagonal parity. To recover the original data, first, substitute the horizontal parity data of disk m into (1) for reversal operations to recover the damaged original disk data; subsequently, employ (2) and (3) to regain the diagonal parity data of disk $m + 1$.

3.2.4. $i<m$, $j<m$. Both the damaged disks retain the encoded original data. To recover, first use the following formula to obtain the S value:

$$S = \left(\bigoplus_{l=0}^{m-2} A[l,m]\right) \oplus \left(\bigoplus_{l=0}^{m-2} A[l,m+1]\right). \quad (6)$$

Subsequently, use the following formulae to obtain Ss as the horizontal blocks $S^{(0)} = S_0^{(0)}, S_1^{(0)}, \ldots, S_{m-1}^{(0)}$ and the diagonal blocks $S^{(1)} = S_0^{(1)}, S_1^{(1)}, \ldots, S_{m-1}^{(1)}$:

$$S_u^{(0)} = \bigoplus_{\substack{l=0 \\ l\neq i,j}}^{m} A[u,l],$$
$$S_u^{(1)} = S \oplus A[u,m+1] \oplus \left(\bigoplus_{\substack{l=0 \\ l\neq i,j}}^{m-1} A\left[\langle u-l\rangle_m, l\right]\right). \quad (7)$$

Here, $S_u^{(0)}$ is the value obtained after an XOR operation on the horizontal blocks of the uth row without the damaged data, while $S_u^{(1)}$ is the value obtained by an XOR operation on the diagonal blocks without the damaged data, $0 \le u \le (m - 1)$. Subsequently, by following the steps below, the lost data in the two damaged disks i and j can be recovered.

(1) Designate the first recovered block and set s to $\langle -(j-i)-1\rangle_m$; subsequently, initialize all the blocks on the $(m - 1)$th row and set $A[m-1,l]$ to 0, where $0 \le l \le (m - 1)$.

(2) Let $A[s,j] = S_{\langle j+s\rangle_m}^{(1)} \oplus A[\langle s+(j-i)\rangle_m, i]$ and $A[s,i] = S_s^{(0)} \oplus A[s,j]$.

(3) Let $s = \langle s-(j-i)\rangle_m$; if $s = (m - 1)$, then stop; otherwise, return to Step (2).

For simplicity, the exact meaning of the application of each formula is not explained in this paper; for more details on the recovery processes, please also refer to the examples in [14, 29, 30].

4. System Architecture and Operation

This section describes methods to employ the fault-tolerant capabilities of EVENODD to avoid file pollution in P2P systems; further, the overheads imposed by the scheme for producing redundant data with respect to the network and storage space are briefly analyzed.

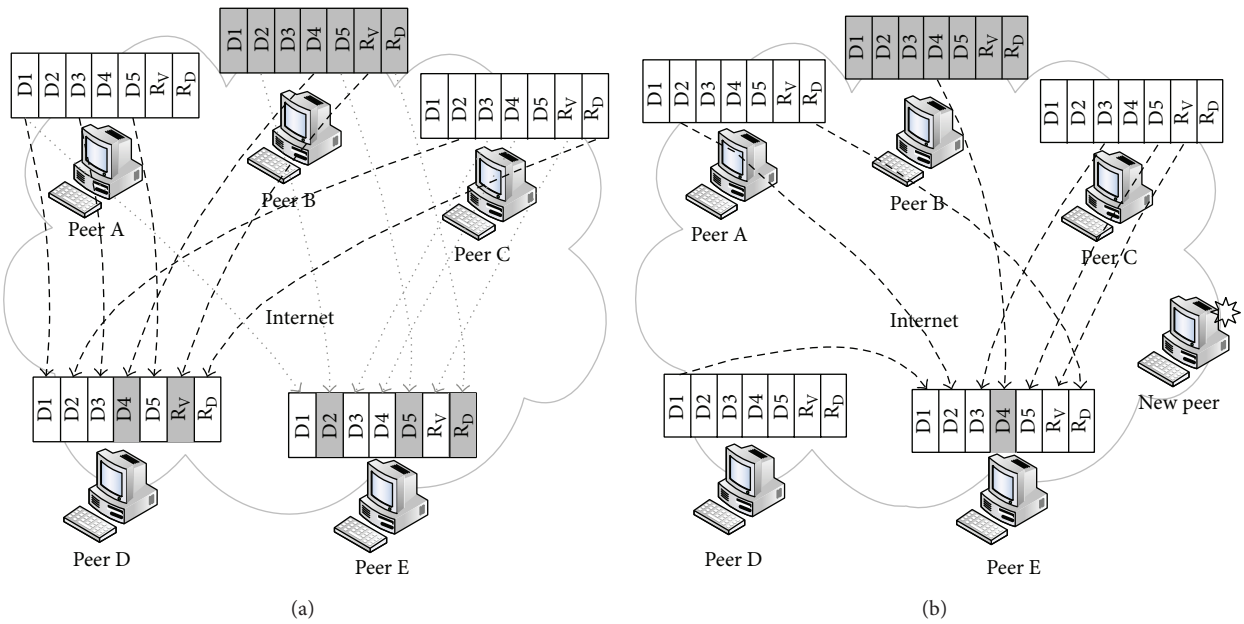

Figure 2: An example of the operational details of the proposed mechanism.

4.1. EVENODD for Antipollution P2P File Sharing. In P2P file sharing systems, in order to efficiently share files, a file will usually be split into several pieces by the P2P file shareware before being placed for sharing [34–36]. Such practices have motivated the adoption of EVENODD to solve pollution problems in P2P file sharing systems. In order to conform with the EVENODD data slicing criteria, a shared data file must be split into m uniformly sized pieces, where m is a prime number.

Figure 2 illustrates an example of the operational details of the proposed mechanism in which five peers—A, B, C, D, and E—network over a P2P file sharing system. As shown in Figure 2(a), A, B, and C have copies of file f, which they all provide to the other peers for downloading. A and C provide good copies, while B provides a polluted copy. Prior to sharing, all of these copies were split into the same number of pieces; in this example, all were split into 5 pieces (m = 5), which, following encoding by EVENODD, resulted in 7 pieces per copy. D and E each place requests to download a copy of f through enquiries to the P2P file sharing system, to which the system responds with the number and locations of the available copies. Owing to the inherent properties of P2P file sharing system, each piece of f downloaded by D and E will generally be from different sources. As can be seen in Figure 2(a), five of the pieces that D and E download separately from A, B, and C are polluted (two by D and three by E). The two polluted pieces downloaded by D are data piece D4 and redundant piece R_v. As this corresponds to situation 2 in Section 3.2, the correct downloaded copy can be recovered by the EVENODD decoding procedure; therefore, peer D has obtained a good copy even though it has downloaded polluted pieces. On the other hand, because E has downloaded more than two polluted pieces, the copy will not be recovered by EVENODD, and therefore E will delete the copy and resend a request to the P2P file system to download a new copy of f. In Figure 2(b), E has obtained a new copy of f after initiating the duplicate download. At this stage, E will download pieces from D as well as from A, B, and C, and since D has become a peer that provides only good copies of f as a result of the previous stage, the probability that E will download good pieces will increase. Thus, only one piece (D4) downloaded by E is polluted in this stage and, as described in Section 3.2, a single polluted data piece can easily be recovered by using (1), (2), and (3) for reversal computation. As a result, E becomes a provider of good copies of f in successive downloads. In accordance with this paradigm, any pollution will be eliminated in a short time.

4.2. Networks and Storage Overhead. After the original data file is successfully split, the previously introduced encoding procedure is employed to produce both the horizontal parity and diagonal parity pieces. As a result of these extra slices, this file has a certain degree of fault-tolerant capabilities. For example, Figure 3 shows an example in which an original data file is split into twelve pieces (A1~D3) that are clustered into four groups (e.g., A1, A2, and A3 are a group). Two pieces of parity data will be created by EVENODD for each group, for example, the two pieces of R_V1 and R_D1 for the first group in Figure 3.

The generated parity data pieces cause a file to occupy more storage space. Assuming a file is split into m pieces, because the generated parity data have a fixed number of two pieces for certain pieces, that is, a group, the higher the m value, the smaller the extra storage space. Figure 4 shows the ratio of the generated parity data size to the original file size as m increases. The extra storage overhead does not reach more than 66% (i.e., the least splitting case when m = 3).

After the EVENODD encoding, a file is disseminated with a certain fault tolerance on the network. Theoretically,

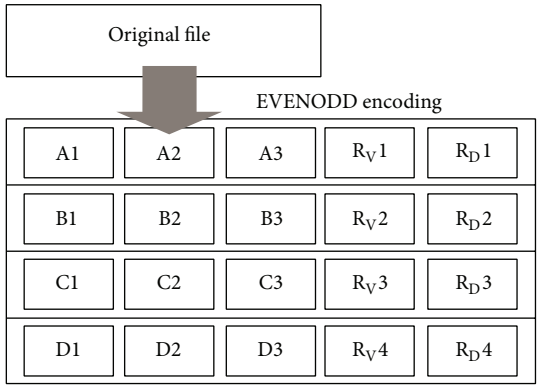

Figure 3: Example of dividing a file into 12 pieces and 4 groups.

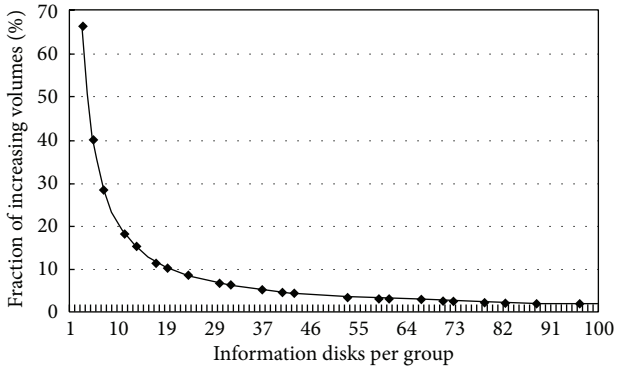

Figure 4: Number of split m and extra storage space.

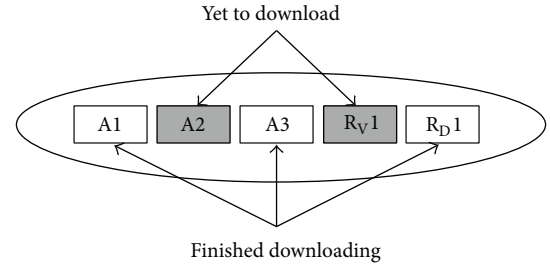

Figure 5: File recovery using EVENODD decoding procedure.

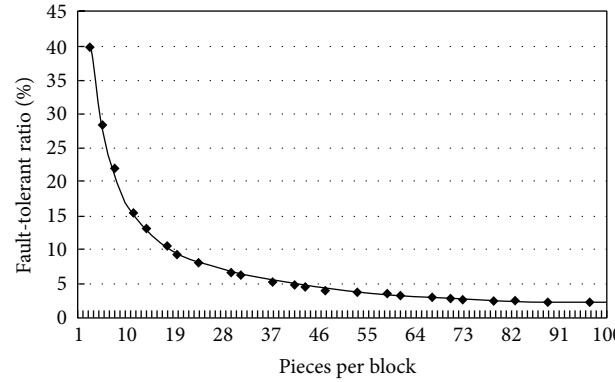

Figure 6: The fault-tolerant rate of EVENODD.

when the shared file size increases, it produces additional network traffic overheads. However, because EVENODD tolerates up to errors in two file pieces, a peer need not download all the pieces of a file to obtain the entire file. The missing two pieces can be recovered by merely applying the EVENODD decoding procedure, as shown in Figure 5.

4.3. Fault-Tolerant Capability. The fault-tolerant capability of a file depends on the total number of pieces split as well as the number of pieces in a group. In the same group, EVENODD could tolerate up to two faulty pieces irrespective of the number of pieces. If a group contains more pieces, the number of groups reduces, and hence, the total number of faults a file can tolerate also reduces. Figure 6 shows the relationship between the number of split pieces and the fault-tolerance capability. For the sake of efficiency, a reasonably small piece size is typically used to split a file in P2P file sharing systems; for example, in the BitTorrent protocol specification [34, 37], the default piece size is 256 KB, but sizes of 512 KB and 1 MB are often observed. FastTrack adopts even smaller piece sizes of 64 KB [35], while in the eMule protocol specification [36], a file piece is called a chunk and has a large size of 9.28 MB; each chunk is then further split into blocks of 180 KB. If a piece size is 512 KB, a movie film of approximately 1 GB comprises approximately 2000 pieces; the fault-tolerance rate for such sizes is not shown in Figure 6 in which only files split up to 97 pieces have been shown (the maximum prime number that is less than 100). In fact, when the number of file pieces reaches 2000, the fault-tolerance rate of the EVENODD will be only 0.1%; this is nearly close to no fault-tolerance capability. One solution for such big files is to split the file first into a reasonable number of blocks and then perform EVENODD encoding, respectively, against each block; however, this could also still enhance fault-tolerance rates. Of course, the space and time required for performing such methods also lead to other issues.

4.4. Tamper Detection. Traditional error detecting codes, such as parity checks and cyclic redundancy checks (CRCs), produce redundant parity bits from the data. These bits are then embedded into the data, and operational data errors are detected by error detecting codes and corrected by forward error correction (FEC) codes. However, if security is not considered, traditional error detecting codes cannot fight purposely devised malicious attacks. In this study, information hiding techniques [38, 39] have been employed for detecting the polluted pieces of files; note that the information hiding concept has been used in many fields, for example, secure communication and multimedia security.

5. System Modeling

In this paper, we have expanded the FM developed by Kumar et al. [15] for the pollution proliferation of P2P file sharing systems and then deduced a model for properly analyzing and evaluating the pollution-avoidance mechanism proposed in

this paper. A brief introduction of FM will be given in this section and the model will be deduced. The various parameters involved in the following modeling and evaluation are summarized in Table 1.

5.1. Fluid Model. The primary goal of FM is to develop a model for P2P file sharing systems; this model should be able to observe, analyze, and evaluate the pollution proliferations and to provide important references for the future designs of antipollution mechanisms. Moreover, it can be used to model and analyze the proliferation of polluted versions and good versions of a title in P2P networks.

Two types of peers are defined in FM: attacking peers that inject polluted copies into P2P systems and benign peers. Let M be the number of benign peers wishing to acquire a title copy; there are several versions of this title, and for each version, there are several available copies and some of them are polluted. The following reasonable assumptions have been made in FM.

(1) When a peer obtains a good version of a title, it stops its search.

(2) If a peer inspects and confirms that a downloaded version has been polluted, it will delete the version and immediately search again for the said title.

(3) If a peer obtains a good version, it will provide this version without any deadlines to other peers on the P2P network for their downloading.

(4) All the peers are homogeneous and exhibit identical behaviors.

In summary, in the FM, there are three types of peers at any instant of time: (1) peers that have a good copy, (2) peers that have a polluted copy, and (3) peers that have no copy. Moreover, any one of the M peers will have a maximum of one copy at any time, regardless of whether it is good or polluted.

In the FM, the inspection time is the time counted from the moment when a user issues a download request to the moment when the user finishes inspecting the fully downloaded file. The average inspection time of a peer is expressed as $1/\mu$, where μ is the inspection rate of a peer, or, in other words, the frequency of the peer deleting a polluted version and issuing a new search request.

Let $\nu(t)$ represent the set of versions (polluted and good ones) appearing on the network at time t. For a version $v \in \nu(t)$, $n_v(t)$ represents the number of copies at time t. Because a peer has a maximum of one copy of any single version at any instant, $n_v(t)$ is also the number of copies of a certain version v at time t.

After a user has enquired for a certain title, he will receive a series of the versions and the number of copies available for each version of the title on P2P networks. Without considering the user behavior for the selection, in general, the probability of a certain specific version v being selected could be modeled as a function of the number of copies of each available version in the system as follows:

$$q_v(t) = f_v\left(n_u(t), u \in \nu(t)\right), \quad v \in \nu(t). \tag{8}$$

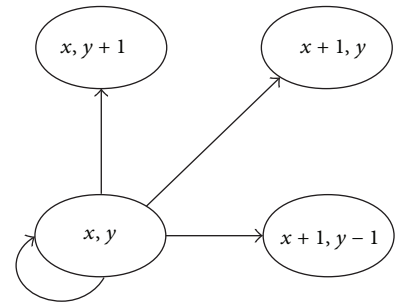

Figure 7: The Markov state transitions for the Copy Centric Model.

Here, $fv(\cdot)$, $v \in \nu(t)$, is an arbitrary function that satisfies the following equation:

$$\sum_{v \in \nu(t)} q_v(t) = 1. \tag{9}$$

The pollution proliferation modeling of the FM could begin from the two extreme cases of the selection distribution $q_v(t)$, $v \in \nu(t)$: the Copy Centric Model and Version Centric Model. In this paper, we have extended the former to deduce the conformable pollution proliferation model. The definition of the Copy Centric Model is that any user will randomly select a copy for download with no particular preferences; that is, each available copy on the network is treated equally. Therefore, the probability of each available version being selected by a user is the ratio of the number of available copies of this version to the total number of available copies; it is given as follows:

$$q_v(t) = \frac{n_v(t)}{\sum_{u \in \nu(t)} n_u(t)}. \tag{10}$$

Let N represent the number of polluted versions that have been placed in the network by an attacker, and let us assume that N is invariable with time. Further, assume that there are M benign peers who all wish to obtain good versions of this title. The FM uses the discrete-state Markov process approach to analyze the pollution proliferation. Figure 7 is such a Markov state chart wherein x and y represent the peers with uncontaminated and polluted copies, respectively, and state transitions occur when a user inspects a downloaded file. As shown in Figure 7, a system can transit from state (x, y) to the following states:

(1) $(x + 1, y)$: a peer without a copy downloads an uncontaminated copy at its first request,

(2) $(x, y + 1)$: a peer without a copy downloads a polluted copy at its first request,

(3) $(x + 1, y - 1)$: a peer with a polluted copy downloads an uncontaminated copy,

(4) (x, y): a peer with a polluted copy downloads a polluted copy at its next request.

Table 1: Summary of parameters involved in the modeling.

Symbol	Description
M	The number of benign peers wishing to acquire a title copy.
μ	The frequency of the peer deleting a polluted version and issuing a new search request.
$\nu(t)$	The set of versions (polluted and good ones) appearing on the network at time t.
v	A version.
$n_v(t)$	The number of copies of a certain version v at time t.
N	The number of polluted versions that have been placed in the network by an attacker.
x	The peers with unpolluted copies.
y	The peers with polluted copies.
$x(t)$	The total number of peers with good copies at a certain time t.
$y(t)$	The total number of peers with polluted copies at a certain time t.
p	The probability that a peer selects a polluted copy and downloads it.
$p(t)$	The probability that a peer selects a polluted copy and downloads it at a certain time t.
m	The number of pieces that a shared file has been split into.

If the rate of inspections is $(M - x)\mu$ at state (x, y), then, according to (10), the probability that a peer selects a polluted copy and downloads it is

$$p = \frac{y + N}{x + y + N}. \tag{11}$$

In order to reduce the complexity of obtaining the probability distribution of time from any initial state to the expected state $(M, 0)$, where M benign peers all obtain good versions of a title, the FM uses the fluid flow approximation for modeling the pollution proliferation model. Let $x(t)$ and $y(t)$ represent the total number of peers with good copies and the total number of peers with polluted copies, respectively, at a certain time t. Based on the Markov model, at any time t, the probability of "a peer that selects a polluted copy and downloads it" becomes

$$p(t) = \frac{y(t) + N}{x(t) + y(t) + N}. \tag{12}$$

When a peer either without a copy or with a polluted copy has downloaded a good copy, the number $x(t)$ of good copies increases. The former will happen at a rate of $[M - x(t) - y(t)]\mu(1 - p(t))$ and the latter at a rate of $y(t)\mu(1 - p(t))$. The fluid equation is thus deduced to

$$\dot{x}(t) = [M - x(t) - y(t)]\,\mu(1 - p(t)) + y(t)\,\mu(1 - p(t)). \tag{13}$$

Similarly, when a peer without a copy downloads a polluted copy, the number $y(t)$ of the polluted copies increases at the rate of $[M - x(t) - y(t)]\mu p(t)$; further, when a peer with a polluted copy downloads a good copy, $y(t)$ will increase at the rate of $y(t)\mu(1 - p(t))$. The fluid equation is thereby deduced to

$$\dot{y}(t) = [M - x(t) - y(t)]\,\mu p(t) - y(t)\,\mu(1 - p(t)). \tag{14}$$

5.2. Modeling with File Pieces. In P2P file sharing systems, assume that there is one version of a shared file, which has been split into m number of pieces. At time t, the probability of a peer downloading a piece of the polluted copy is given by $p(t)$ in (12). However, for $x(t)$ at time t, for either a peer without a copy or a peer with a polluted copy downloading a good copy, the former happens at a rate of speed increase of $(M - x(t) - y(t))\mu(1 - P(t))^m$, while the latter happens at $y(t)\mu(1 - P(t))^m$. Hence, we deduce a new $\dot{x}(t)$ as follows:

$$\begin{aligned}\dot{x}(t) = {} & (M - x(t) - y(t))\,\mu(1 - p(t))^m \\ & + y(t)\,\mu(1 - p(t))^m.\end{aligned} \tag{15}$$

However, deducing a new $\dot{y}(t)$ is more complicated because, to a user, if a polluted piece is found in a downloaded file, the file is seen as polluted, and this makes the number of all possible polluted pieces a critical argument in deducing a new $\dot{y}(t)$. Considering the growth of $y(t)$, to a peer that does not own any piece, the probability that "it will download more than one polluted piece from other peers" is as follows:

$$\begin{aligned}& p(t)^m + C_1^m \times p(t)^{m-1} \times (1 - p(t))^1 \\ & \quad + C_2^m \times p(t)^{m-2} \times (1 - p(t))^2 \\ & \quad + \cdots \\ & \quad + C_{m-1}^m \times p(t)^1 \times (1 - p(t))^{m-1}.\end{aligned} \tag{16}$$

This expression includes the cases from "all downloaded pieces are polluted," "only one downloaded piece is not polluted," and "two downloaded pieces are not polluted" to "all downloaded pieces are not polluted." On the other hand, to a peer that has always downloaded the polluted piece, when it requests again and then downloads a good copy, the speed

of increase in $y(t)$ is $y(t)\mu(1-p(t))^m$. By substitution, one can obtain the new $\dot{y}(t)$ as follows:

$$\begin{aligned}\dot{y}(t) = {} & \big(M - x(t) - y(t)\big)\mu \\ & \times \Big\{p(t)^m + C_1^m \times p(t)^{m-1} \times (1-p(t))^1 \\ & \quad + C_2^m \times p(t)^{m-2} \times \big(1-p(t)\big)^2 \\ & \quad + \cdots \\ & \quad + C_{m-1}^m \times p(t)^1 \times \big(1-p(t)\big)^{m-1}\Big\} \\ & - y(t)\,\mu\big(1-p(t)\big)^m.\end{aligned} \tag{17}$$

5.3. Modeling with EVENODD. In order to simplify the EVENODD modeling process, assume that a file is split into m pieces and the file, after being encoded, will have two pieces more than the original file. First, consider the probability that "a peer that owns no any piece downloads more than three polluted pieces," which is a series of permutations and combinations and is expressed as $p_y(t)$. The equation is as follows:

$$\begin{aligned}p_y(t) = {} & p(t)^{m+2} + C_1^{m+2} \times p(t)^{m+1} \times (1-p(t))^1 \\ & + C_2^{m+2} \times p(t)^m \times \big(1-p(t)\big)^2 \\ & + \cdots \\ & + C_{m-1}^{m+2} \times p(t)^3 \times \big(1-p(t)\big)^{m-1}.\end{aligned} \tag{18}$$

In this equation, since the file itself can perform self-recovery from up to two polluted pieces, it is not necessary to consider the probability of the cases of "downloading one polluted file piece" and "downloading two polluted file pieces."

Now, let us consider the case in which a peer owns a polluted copy and deduce the probability $p_x(t)$ that "it requests again and downloads a good copy." This is deduced and shown as follows:

$$\begin{aligned}p_x(t) = {} & (1-p(t))^{m+2} + C_1^{m+2} \times (1-p(t))^{m+1} \times p(t) \\ & + C_2^{m+2} \times \big(1-p(t)\big)^m \times p(t)^2.\end{aligned} \tag{19}$$

In the case of a file that has been encoded with EVENODD, a user can immediately stop downloading as long as there are m unpolluted pieces among the downloaded ones; this is because this file can be recovered by using the EVENODD decoding procedure. However, this will make the summation of $p_x(t)$ and $p_y(t)$ greater than 1, which is theoretically unreasonable. Nevertheless, the assumption that a peer will stop downloading after having downloaded all the pieces of a file is still reasonable. The first item of the latter two items is the probability of "requesting a download again and acquiring one polluted piece", and the second item is the probability of "requesting a download again and getting two polluted pieces". The latter two cases will cause no harm because the file can be recovered by using the EVENODD decoding procedure. $\dot{x}(t)$ and $\dot{y}(t)$ can then be deduced as follows, respectively:

$$\dot{y}(t) = \big(M - x(t) - y(t)\big)\mu p_y(t) - y(t)\,\mu p_x(t), \tag{20}$$

$$\dot{x}(t) = \big(M - x(t) - y(t)\big)\mu p_x(t) + y(t)\,\mu p_x(t). \tag{21}$$

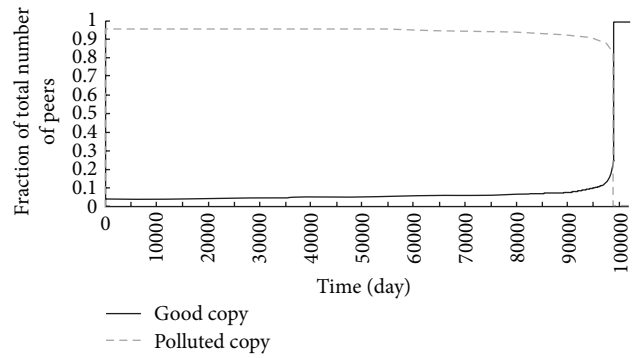

Figure 8: Pollution proliferation without EVENODD.

6. Evaluation

In this section, we present our evaluation results of the antipollution mechanism proposed in this paper. The evaluation was performed using two approaches—by modeling and by simulations.

6.1. Evaluation by Modeling. It is difficult to estimate the distributions of $x(t)$ and $y(t)$ in (15) and (17), respectively, by inspecting the equations in Section 5.2; numerical analysis is used to acquire the approximation solutions of the formulas. Let $x(0) = 1000$, $N = 1000$, $m = 5$, $\mu = 1$ (query/day), and $M = 100000$ be the initial values. Substituting them into (15) and (17) with time varying beyond a major swing in the number of peers that own good or polluted copies yields the result shown in Figure 8; here, the dotted and darker curves represent the number of polluted and good copies, respectively, normalized as a fraction of the total number of peers. From this figure, peers that have downloaded a polluted piece are barely able to obtain a good copy for a very long time. This result is quite predictable because the probability is very strict for a peer to obtain a good and complete file for large m values. For a popular file, if an attacker produces false peers that are close in number to the number of peers that own good copies when it has just begun disseminating, the attack would become very threatening.

Similarly, it is difficult to estimate the distributions of $x(t)$ and $y(t)$ from (20) and (21), respectively; in fact, these equations are even more complicated than (15) and (17). An identical numerical analysis approach is used to deduce the approximation solutions of these formulae by setting the initial values for $x(0)$, N, m, μ, and M identical to those for (15) and (17). The result is shown in Figure 9; the light gray and darker curves represent the number of polluted and good copies, respectively. It is obvious that these curves fall and rise continuously and quickly from the very beginning and converge to become flat at approximately

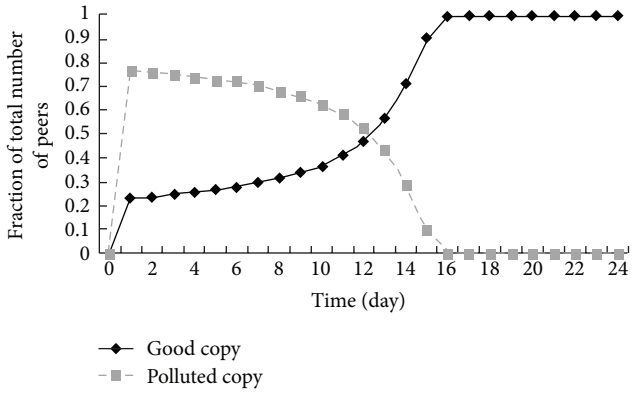

Figure 9: Pollution proliferation with EVENODD.

the 16th day as compared to the more than 100000 days in Figure 8. This means that when the attacker begins to disseminate polluted copies, the proposed mechanism begins to produce a suppression effect, and the pollution attack could be exterminated on the 16th day.

Although there is a large difference between the convergence times in Figures 8 and 9, the curve patterns of these two figures are similar. The initial values of both the environment parameters $x(0)$ and N being equal to 1000 imply that, for each file piece, the probabilities that, "on the first day, a peer has downloaded a polluted piece" and, "on the first day, a peer has downloaded a unpolluted piece" are identical. However, when a single piece downloaded by a peer is polluted, the downloaded file will be treated as a polluted one. As a result, on the first day, the number of polluted copies is higher than that of the good copies. Under the assumptions that a peer who owns a complete and good copy will share this file indefinitely and the peers who have downloaded polluted files will request to download again continuously until they receive a complete and good copy, the number of polluted copies will decrease and the number of good copies will increase with time. In this manner, adopting the EVENODD procedure that tolerates two polluted pieces downloaded by a peer clearly makes the convergence speed in Figure 9 faster than that in Figure 8.

6.2. Evaluation by Simulations. The evaluation results in the previous section are obtained from the equations with many postulations. In this section, we describe the simulation approach that was used to verify the accuracies of the obtained results. Several simulation experiments were conducted to evaluate the use of EVENODD in suppressing and eliminating pollution attacks in P2P file sharing systems. In the experiments, PCs (3.0 GHz processor; 2 GB DDR II 800 RAM) were used, and Visual C++ was selected as the programming tool.

6.2.1. Behavior of Nodes. In P2P file sharing systems, a peer must act and download before possibly receiving any polluted file. Therefore, the focus of this simulation experiment was on peer behaviors. General impact factors such as network traffic, framework, and search time that are considered in the simulation experiments of P2P file sharing networks were not be considered in this experiment. The conditions and assumptions of the simulation environments of this experiment were as follows.

(1) The number of peers that share a complete and healthy file on the network, the number of benign peers that demand this file, and the number of attacking peers are fixed and maintained constant from the beginning of sharing the file.

(2) None of the peers leave the network.

(3) A peer that has successfully downloaded a certain file definitely shares all the pieces of this file with the peers that have not completed downloading.

These assumptions are in accordance with the modeling criteria described in the previous section. In addition, the procedure followed by a benign peer for requesting a file is fairly simple and is as follows.

(1) Select a file piece that is still missing.

(2) Choose a peer that owns the piece and will provide the benign peer with the piece.

(3) Check whether the requested file has been completely downloaded, and if so, stop selecting pieces and peers.

6.2.2. Simulation Results. On the basis of the conditions and assumptions and setting the initial vales of $x(0) = 100$, $N = 100$, $m = 5$, $\mu = 1$ (query/day), and $M = 2000$, the first experiment was conducted on the pollution evolution in P2P file sharing systems that do not use EVENODD. The results are shown in Figure 10. When a P2P file sharing network was attacked, as much as 92% of the peers were polluted from the very beginning. On the 4th day, the number of polluted peers decreased to approximately 80%, whereas after the 4th day, the number of polluted peers significantly decreased. Moreover, on the 10th day, all the peers on the file sharing network obtained a good copy of the file. Since, at time t, a peer can only own a copy of a file, the number of polluted peers was also the number of polluted copies. Moreover, because of the assumption that a peer that owns a good copy of a file will definitely share the file with other peers and the peers that have realized that they have downloaded a polluted copy will immediately request a new download, all the peers in the system ultimately received good copies.

Figure 11 presents the experimental results from the EVENODD procedure on a P2P file sharing network; the number of polluted copies decreased fast after the first day, and on the 4th day, all the peers in the P2P file sharing network obtained a good copy of a file. In comparison with Figure 10, this experimental result confirms that the usage of EVENODD enables the efficient suppression and removal of file pollution in a shorter time on a P2P file sharing network; the convergence speed is approximately 40%~60% faster.

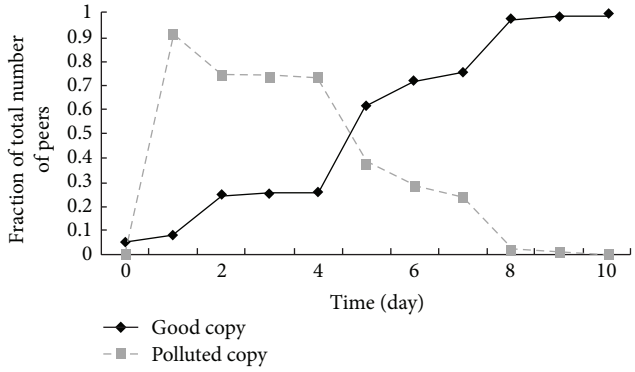

Figure 10: Pollution proliferation without EVENODD.

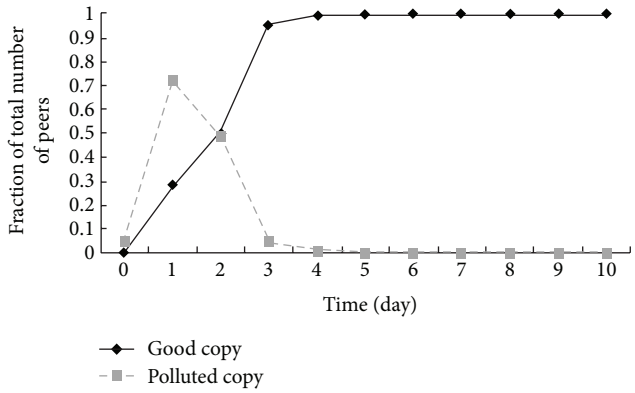

Figure 11: Pollution proliferation with EVENODD.

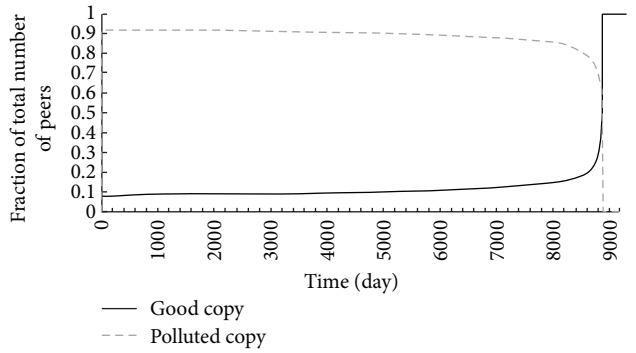

Figure 12: Pollution proliferation in a small-scale network without EVENODD.

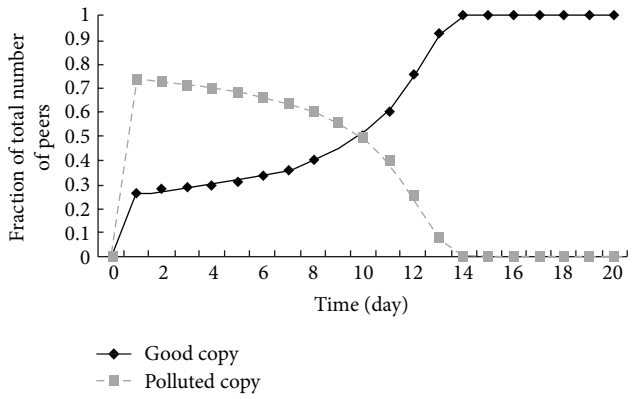

Figure 13: Pollution proliferation in a small-scale network with EVENODD.

7. More Evaluation by Modeling

The accuracy of the model developed in Section 5.3 can be demonstrated by comparing the modeling evaluation results with those of simulation in Section 6. In this section, additional evaluation of the proposed model mechanism is conducted. In this further evaluation, two kinds of network scale settings are used: a large-scale network, as described in Section 6.1, in which $x(0) = 1000$, $N = 1000$, $m = 5$, $\mu = 1$ (query/day), and $M = 100000$; a small-scale network, as described in Section 6.2, with $x(0) = 100$, $N = 100$, $m = 5$, $\mu = 1$ (query/day), and $M = 2000$.

7.1. Amending Equations. In Sections 6.1 and 6.2, the initial environmental parameter configurations were different; they were for large-scale and small-scale networks, respectively. In order to understand the precision of the experiments described in Section 6.2, the settings in small-scale network are used to redo the calculations in Section 6.1. The results are shown in Figure 13.

From Figures 11 and 13, which are the evaluation results of the model and simulation experiment, respectively, the pollution situations can be observed to be continuingly decreasing from the beginning of the first day. However, the simulation results show that the file pollution in the system was totally eliminated by the 4th day, while the model results show that the elimination happened only by the 16th day. Similar results were obtained when comparing the results of Figures 10 and 12, and here, the differences in the durations of the pollution convergence were even larger. This implies the deduced model differs from the practical one. The reasons for such a situation may be obtained by examining the following loopholes existing during the model deduction.

(1) In the model analysis, time is discrete (day by day). As a result, each peer sees the previous day's distributions of the copies in the P2P file sharing network.

(2) If a peer has downloaded a polluted piece, the entire file containing the piece will be considered as polluted regardless of the presence of other unpolluted file pieces, and this peer will also be considered a polluted peer.

(3) Similar to (2), as long as a peer has downloaded a piece from a peer that is believed to be polluted, it will be seen as a polluted peer, even if the downloaded piece is a good one.

In order to deduce a model that is more realistic, a new assumption will be added—a peer will delete all the polluted file pieces that were previously downloaded before requesting

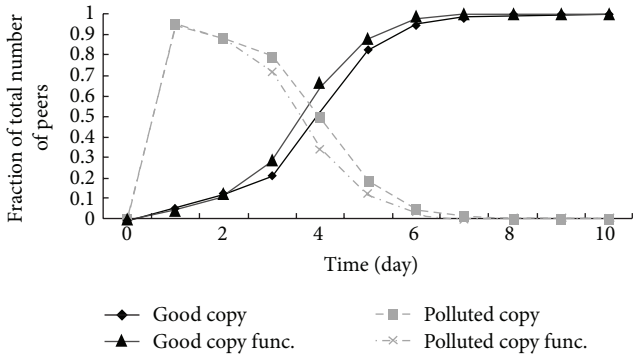

Figure 14: The pollution proliferation without using EVENODD after modifying the assumption.

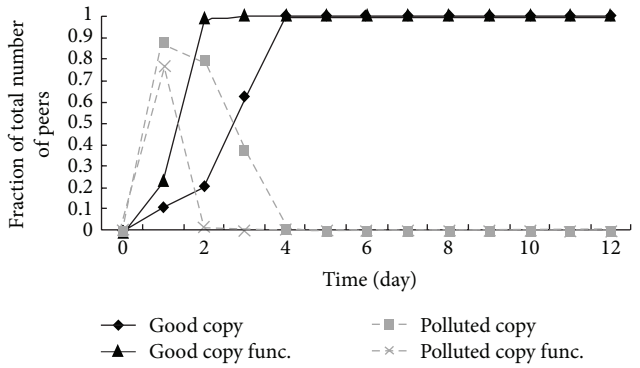

Figure 15: The pollution proliferation that uses EVENODD after modifying the assumption.

a download again. Under this assumption, the probability that a peer will obtain a polluted piece is modified as follows:

$$p(t) = \frac{N}{x(t) + N}. \tag{22}$$

Based on the modified assumptions, the evaluations of Sections 6.1 and 6.2 are performed again to yield the results shown in Figures 14 and 15, which are the evaluation results of a P2P file sharing system adopting and not adopting EVENODD, respectively. In these two figures, both the results of model evaluations and simulation experiments have been deliberately combined for comparisons, and from them, it is obvious that the model with the modified assumptions has a curved shape similar to the simulations of the practical situation.

Furthermore, the assumption in the Fluid Model that if a peer obtains a good version it will provide it without deadlines to other peers for downloading is also unrealistic. Several types of peer that will be reluctant to share downloaded files exist, including free riding peers as well as those that leave the P2P file sharing system naturally. To account for this, a new parameter P_S is added to represent the probability that a peer obtaining a good version will share it indefinitely. The parameter P_S is assumed to be an average value and will

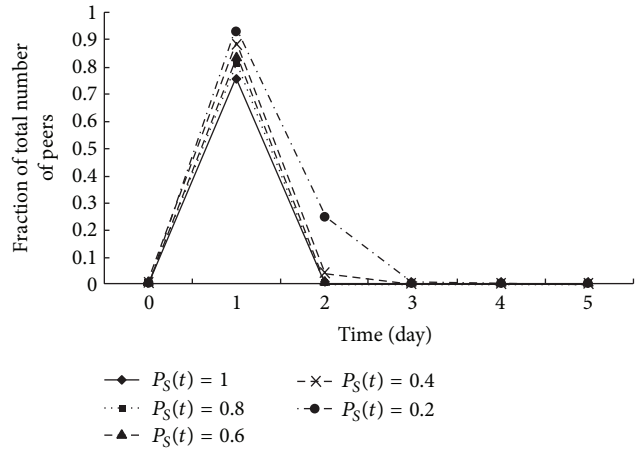

Figure 16: The effects of P_S on the elimination of polluted copies for values of P_S of 1, 0.8, 0.6, 0.4, and 0.2.

not be changed during the process of pollution elimination. Using P_S, (21) can be modified as follows:

$$\dot{x}(t) = \left[\left(M - x(t) - y(t)\right)\mu p_x(t) + y(t)\,\mu p_x(t)\right] * P_S. \tag{23}$$

Because P_S has no effect on $\dot{y}(t)$, (20) stays the same; the evaluation in the following subsections is performed using (20) and (23). Figure 16 shows the effects of P_S on the functioning of the proposed mechanism for values of P_S of 1, 0.8, 0.6, 0.4, and 0.2.

The curves in Figure 16 show variation in the number of polluted copies; as can be seen, although P_S has an impact on the mechanism, pollution is still eliminated in about two or three days. This demonstrates that the proposed mechanism can efficiently avoid pollution.

7.2. Impact of Network Scale. In this subsection, we investigate the impact of network scale on the convergence time of pollution in the proposed mechanism. While a comparison of Figures 8 and 13 would suggest that there is an effect on convergence time owing to network scale, the proportional relation among M, N, and $X(0)$ in the model would seem to contravene this. To check this, a new parameter R can be added to represent the relations between any two parameters. R is defined as

$$R_{A\text{-}B} = \frac{\text{Value of parameter } A}{\text{Value of parameter } B}. \tag{24}$$

In the large-scale network setting, for example, $R_{M\text{-}N}$ will equal 100, as $M = 100000$ and $N = 1000$. For both the large- and small-scale network settings, the values of N and $X(0)$ were varied in order to fix $R_{M\text{-}N}$ and $R_{M\text{-}X(0)}$ at the same values in both scale networks. The results of this experiment are given in Table 2, which shows a day-by-day breakdown of the proportion of good copies in each network.

It can be seen that the results in columns 2 and 4 and columns 3 and 5 of Table 2, respectively, are identical. Based on this, it would appear that network scale has no effect on

Table 2: The variation of good copies in large and small network.

Time (day)	$M = 100000$		$M = 2000$	
	$N = 2000,\ X(0) = 2000$ $R_{M-N} = 50,\ R_{M-X(0)} = 50$	$N = 2000,\ X(0) = 1000$ $R_{M-N} = 50,\ R_{M-X(0)} = 100$	$N = 40,\ X(0) = 40$ $R_{M-N} = 50,\ R_{M-X(0)} = 50$	$N = 40,\ X(0) = 20$ $R_{M-N} = 50,\ R_{M-X(0)} = 100$
0	0.168152823	0.037562227	0.168152823	0.037562227
1	0.952718092	0.429983234	0.952718092	0.429983234
2	0.999632505	0.995571964	0.999632505	0.995571964
3	0.999941027	0.999822007	0.999941027	0.999822007
4	0.999982692	0.999961957	0.999982692	0.999961957
5	0.99999384	0.999987828	0.99999384	0.999987828
6	0.999997645	0.99999553	0.999997645	0.99999553
7	0.999999075	0.999998271	0.999999075	0.999998271
8	0.999999632	0.999999317	0.999999632	0.999999317
9	0.999999853	0.999999728	0.999999853	0.999999728
10	0.999999941	0.999999892	0.999999941	0.999999892

the proposed mechanism as long as the ratios between M and N and M and $X(0)$, respectively, are the same in both scale networks.

7.3. Impact of Fraction of Initial Good Copies and Polluted Copies. This subsection conducted an experiment to inspect the pollution attack resistance ability of a P2P system that uses EVENODD. The aim was to verify the impacts on the convergence times caused by the variance of the initial values $X(0)$ and N. A parameter K that used to represent the variance is defined as

$$K = \frac{N}{X(0)}. \tag{25}$$

A larger value of K represents a fiercer pollution attack by the attacker in the beginning. The experimental result is shown in Figure 17, the pollution convergence time increased with K in both large and small-scale networks. However, in P2P file sharing systems that use EVENODD, the increase in K has an obviously smaller impact on the convergence speed—only approximately 1/10th that of the P2P file sharing systems without EVENODD. This experiment confirms that using EVENODD to fight against and remove file pollution attacks in P2P file sharing networks is effective and robust.

7.4. Impact of Degree of Pollution Attack. The parameters R and K both indicate the degree of pollution attack; for example, a larger value of K or a smaller value of $R_{M\text{-}N}$ represents a more fierce attack. In this subsection, an experiment to inspect the impact of the number of split pieces on the convergence time of pollution is described. This assessment is divided into two parts, one for $K = 1$ and another for $K = 2$. In the first part, parameters of a large-scale network are adopted—$M = 100000$, $\mu = 1$ (query/day), and $P_S = 0.6$—and the setting of $K = 1$ implies that $N = X(0)$ and also that $R_{M\text{-}N} = R_{M\text{-}X(0)}$. The first part of the experiment was performed using three sets each of N and $X(0)$, with the results shown in Figure 18. In this figure, the curves represent the proportion of polluted copies in the P2P file sharing system.

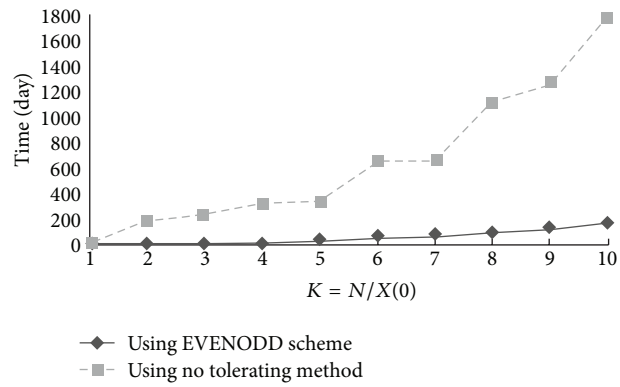

Figure 17: The resistance ability of a P2P system a subject to pollution attack.

Figure 18(c) shows the results of a more fierce attack. As can be seen, the larger number of split pieces results in the slower elimination of polluted copies, as the pollution attack will be more severe in the beginning. Next, an experiment similar to the first part was conducted, with the results shown in Figure 19.

Figure 19 shows the same results as in Figure 18; this is to be expected, as the analysis in Section 4.3 determined that increasing the number of split pieces will lower the fault tolerance capability of EVENODD. As will be discussed in the next subsection, a more fierce attack will present challenges for the proposed mechanism not only in terms of efficiency but also in terms of network and storage overhead.

7.5. Comparison. The analytical models in the classic antipollution mechanisms Scrubber [6], Credence [12], and Blacklist [13] are implemented and compared with the proposed mechanism in terms of convergence time. In this evaluation, the large-scale network settings described in Section 6.1 are used. Parameters specific to Scrubber and Credence adopted the settings from [6], but the user feedback and β are set to 0.8 and 1.0, respectively, for a more realistic assessment. Because

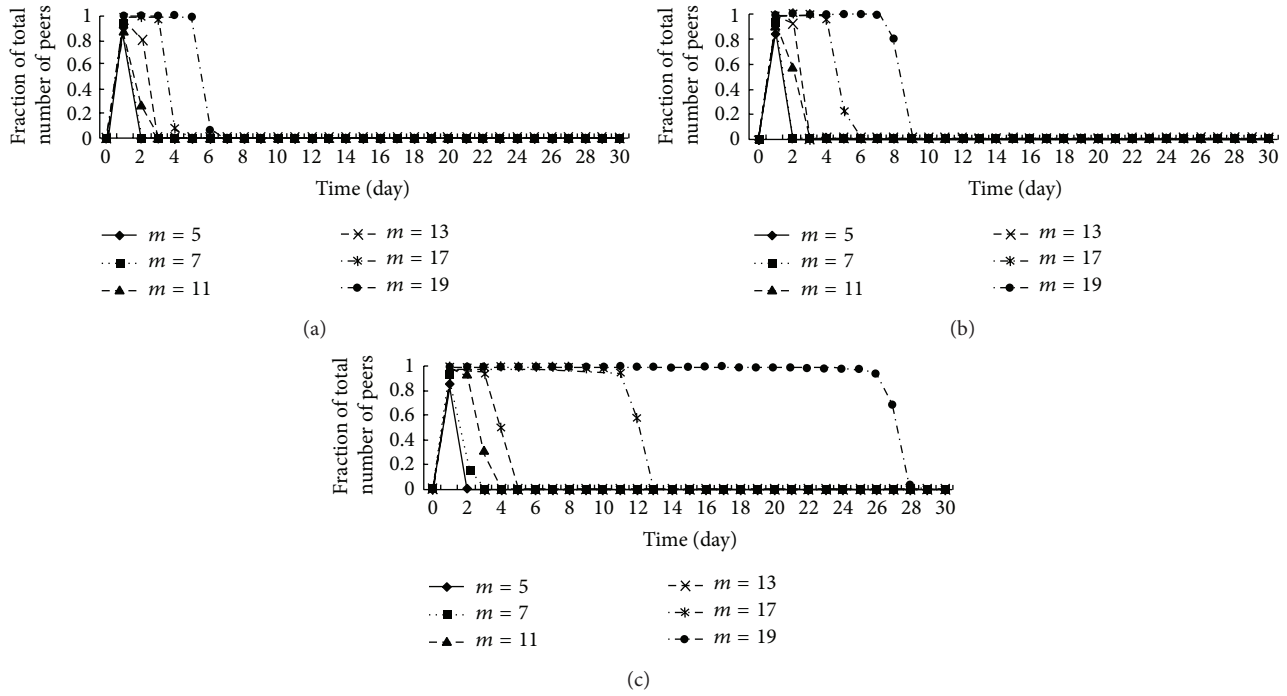

Figure 18: Elimination speed of polluted copies for values of m of 5, 7, 11, 13, 17, and 19 in a large-scale network—$M = 100000$, $\mu = 1$ (query/day), $P_S = 0.6$, and $K = 1$: (a) $N = 100$, $X(0) = 100$; (b) $N = 200$, $X(0) = 200$; (c) $N = 1000$, $X(0) = 1000$.

the derivation of the analytical model in this paper is based on the Fluid Model [15], parameters that are specific to Blacklist adopted the settings from that paper. The experiment results are shown in Figure 20.

7.6. Discussion. Most research to date focusing on the development of pollution countering measures has adopted some sort of reputation system based on a globally agreed-upon voting scheme. However, such systems have several inherent weaknesses. First, reputation systems face the cold start problem [26] as, for example, in the case where a new user will have to belong to the system for a period of time in order to build up sufficient reputation. Second, because reputation systems are vulnerable to malicious attacks [40] such as the Sybil attack, additional defensive solutions must be invented and inserted into the system. Third, users may avoid voting or provide a false vote, a tack taken deliberately by malicious users and accidentally by good users who forget to vote or mistakenly cast a reverse vote. Fourth, a reputation system must usually be accompanied by an incentive mechanism [41] that encourages users to provide truthful feedback while also allowing good users to recover from damaged reputation. Together, these potential pitfalls make existing pollution countering schemes based on reputation complicated and difficult to implement.

The major drawback of the proposed mechanism lies in its requirement for additional network and storage overhead, with the analysis in Section 4.2 showing that overhead factors increase as the number of split pieces decreases. However, the experiment conducted in Section 7.4 indicates results contrasting with these in terms of efficiency. Instead of proposing an appropriate number of split pieces, this paper proposes that different types of shared file should be split into different numbers of pieces. For example, a large file split into a small number of pieces will result in high network and storage overhead; furthermore, document files are suited to being broken into smaller numbers of file pieces because they are generally small and their data integrity is important. Another defect of the proposed mechanism comes from the overhead incurred by the tamper detection scheme. When no tamper detection measures are imposed on the proposed mechanism, a user must inspect a downloaded file manually, which will increase the time used for downloading a file and the burden of pollution elimination. On the other hand, if a tamper detection technique as described in Section 4.4 is adopted, the proposed mechanism will become more complicated and cause more network and storage overhead (in the form of additional parity data, e.g.).

The proposed mechanism is very simple and requires only the implementation of EVENODD encoding and decoding procedures on each peer node of a P2P file sharing system. Moreover, as shown in the experimental results, pollution can be effectively suppressed whether or not users can discover and delete polluted files in a timely manner. If reputation-based pollution countering measures can be used in conjunction with the proposed mechanism, file pollution can be eliminated even more effectively.

7.7. Future Work. In this study, we successfully derived an analytical model and obtained numerical results within only a limited space owing to numerous assumptions. However, the analysis of the proposed mechanism can be more realistic if

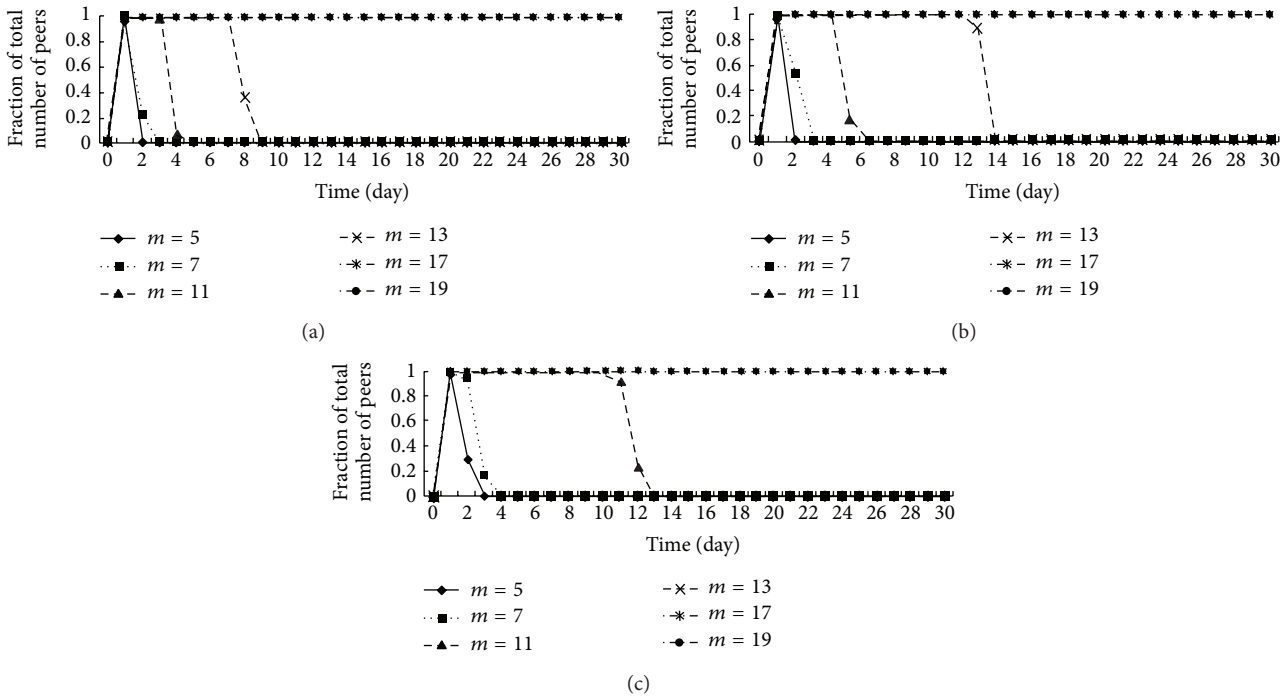

FIGURE 19: Elimination speed of polluted copies for values of m of 5, 7, 11, 13, 17, and 19 in a large-scale network—$M = 100000$, $\mu = 1$ (query/day), $P_S = 0.6$, and $K = 2$: (a) $N = 100$, $X(0) = 50$; (b) $N = 200$, $X(0) = 100$; (c) $N = 1000$, $X(0) = 500$.

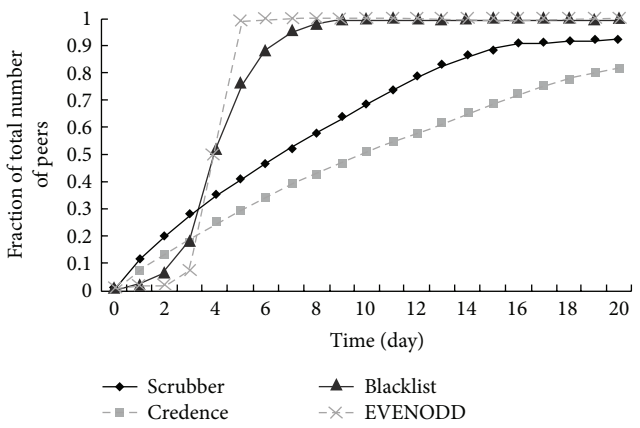

FIGURE 20: Convergence time of Scrubber, Credence, Blacklist, and the proposed mechanism.

some of these assumptions are not made. First, we consider that the number of polluters is varied. Polluters who are in the system may collude with more malicious peers than those who are not in the system in order to launch a more fierce attack but may give up and exit the system due to the presence of an antipollution mechanism. This variable can be taken into consideration in the derivation of the probability that a peer selects a polluted copy and downloads it (see (22)). Second, since a peer can join or leave a P2P file sharing system at any time, the number of peers is varied from the beginning of file sharing. This includes two variables: one is the probability that a peer stops sharing a downloaded file and leaves the system after finishing the download and the other is that new peers who query for the file join the system in each round. In the former, the files downloaded by peers who stop sharing the file after downloading can be polluted or unpolluted. This implies that the number of polluted and unpolluted versions can be reduced. As in the case of the latter, not only benign peers but also polluters can join the system.

8. Conclusion

Pollution attacks are conducted in P2P file sharing systems to stop the downloading of copyrighted products; further, in recent years, many researchers have attempted to develop mechanisms for legally downloading copyrighted products. We believe that even if the dispute over copyrighted product dissemination ends in the near future, file pollution attacks might become a tool for malicious users to attack P2P file sharing systems, and moreover, this will seriously impact the efficiency of P2P file sharing systems and even degrade the network effectiveness. Hence, we have attempted establish a mechanism to suppress and eliminate the pollution attacks in P2P file sharing systems. By sectioning a file into suitable number of pieces and by applying EVENODD to these pieces, the proposed approach has successfully achieved this goal. Both the correctness and efficiency of the approach were verified by simulation experiments; the evaluation results demonstrate that the application of EVENODD to a P2P file sharing system could effectively reduce 40 to 60% of the pollution duration. Further, EVENODD could be easily embedded in the present P2P file sharing software without requiring any changes to its underlying infrastructures.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] N. S. Good and A. Krekelberg, "Usability and privacy: a study of Kazaa P2P file-sharing," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03)*, pp. 137–144, Lauderdale, Fla, USA, April 2003.

[2] S. Sen and J. Wang, "Analyzing peer-to-peer traffic across large networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 2, pp. 219–232, 2004.

[3] Y. K. Kwok, "Autonomic peer-to-peer systems: incentive and security issues," in *Autonomic Computing and Networking*, pp. 205–236, Springer, New York, NY, USA, 2009.

[4] J. Liang, R. Kumar, Y. Xi, and K. W. Ross, "Pollution in P2P file sharing systems," in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '05)*, vol. 2, pp. 1174–1185, Miami, Fla, USA, March 2005.

[5] N. Christin, A. S. Weigend, and J. Chuang, "Content availability, pollution and poisoning in file sharing peer-to-peer networks," in *Proceedings of the 6th ACM Conference on Electronic Commerce (EC '05)*, pp. 68–77, Vancouver, Canada, June 2005.

[6] C. Costa, V. Soares, J. Almeida, and V. Almeida, "Fighting pollution dissemination in peer-to-peer networks," in *Proceedings of the ACM Symposium on Applied Computing (SAC '07)*, pp. 1586–1590, Seoul, Republic of Korea, March 2007.

[7] F. Benevenuto, C. Costa, M. Vasconcelos, V. Almeida, J. Almeida, and M. Mowbray, "Impact of peer incentives on the dissemination of polluted content," in *Proceedings of the ACM Symposium on Applied Computing (SAC '06)*, pp. 1875–1879, Dijon, France, April 2006.

[8] S. Shin, J. Jung, and H. Balakrishnan, "Malware prevalence in the Kazaa file-sharing network," in *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement (IMC '06)*, pp. 333–338, Rio de Janeiro, Brazil, October 2006.

[9] J. Liang, N. Naoumov, and K. W. Ross, "The index poisoning attack in P2P file sharing systems," in *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM '06)*, pp. 1–12, Barcelona, Spain, April 2006.

[10] P. Rodriguez, S. M. Tan, and C. Gkantsidis, "On the feasibility of commercial, legal P2P content distribution," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 75–78, 2006.

[11] W. Dong, S. B. Yang, and X. Q. Liu, "Artificial immunology based anti-pollution P2P file sharing system," in *Proceedings of the 6th International Conference on Grid and Cooperative Computing (GCC '07)*, pp. 82–87, Los Alamitos, Calif, USA, August 2007.

[12] K. Walsh and E. G. Sirer, "Fighting peer-to-peer spam and decoys with object reputation," in *Proceedings of the ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems*, pp. 138–143, Philadelphia, Pa, USA, August 2005.

[13] J. Liang, N. Naoumov, and K. W. Ross, "Efficient blacklisting and pollution-level estimation in P2P file-sharing systems," in *Technologies for Advanced Heterogeneous Networks*, vol. 3837 of *Lecture Notes in Computer Science*, pp. 1–21, Springer, Berlin, Germany, 2005.

[14] M. Blaum, J. Brady, J. Bruck, and J. Menon, "EVENODD: an efficient scheme for tolerating double disk failures in raid architectures," *IEEE Transactions on Computers*, vol. 44, no. 2, pp. 192–202, 1995.

[15] R. Kumar, D. D. Yao, A. Bagchi, K. W. Ross, and D. Rubenstein, "Fluid modeling of pollution proliferation in P2P networks," in *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems*, pp. 335–346, June 2006.

[16] P. Dhungel, X. Hei, K. W. Ross, and N. Saxena, "The pollution attack in P2P live video streaming: measurement results and defenses," in *Proceedings of the Workshop on Peer-to-peer Streaming and IP-TV (P2P-TV '07)*, pp. 323–328, Kyoto, Japan, August 2007.

[17] D. Dumitriu, E. Knightly, A. Kuzmanovic, I. Stoica, and W. Zwaenepoel, "Denial-of-service resilience in peer-to-peer file sharing systems," *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, no. 1, pp. 38–49, 2005.

[18] C. Shi, D. Han, X. Hu, and Y. Yu, "A unified model of pollution in P2P networks," in *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing Symposium (IPDPS '08)*, pp. 1–12, Miami, Fla, USA, April 2008.

[19] R. Thommes and M. Coates, "Epidemiological modelling of peer-to-peer viruses and pollution," in *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM '06)*, vol. 6, pp. 1–12, Barcelona, Spain, April 2006.

[20] Q. Gu, K. Bai, H. Wang, P. Liu, and C. H. Chu, "Modeling of pollution in P2P file sharing systems," in *Proceedings of the 3rd IEEE Consumer Communications and Networking Conference (CCNC '06)*, pp. 1033–1037, Las Vegas, Nev, USA, January 2006.

[21] U. Lee, M. Choi, J. Cho, M. Y. Sanadidi, and M. Gerla, "Understanding pollution dynamics in P2P file sharing," in *Proceedings of the International Workshop on Peer-to-Peer Systems*, vol. 6, pp. 1–6, Santa Barbara, Calif, USA, 2006.

[22] S. Yang, H. Jin, B. Li, and X. Liao, "A modeling framework of content pollution in peer-to-peer video streaming systems," *Computer Networks*, vol. 53, no. 15, pp. 2703–2715, 2009.

[23] W. Dong, S. B. Yang, and L.-T. Guo, "Object reputation based anti-pollution P2P file sharing system," in *Proceedings of the 1st International Conference on Digital Information Management*, pp. 538–543, Bangalore, India, December 2006.

[24] Z. Cai, R. Chen, J. Feng, C. Tang, Z. Chen, and J. Hu, "A holistic mechanism against file pollution in peer-to-peer networks," in *Proceedings of the 24th Annual ACM Symposium on Applied Computing (SAC '09)*, pp. 28–34, Honolulu, Hawaii, USA, March 2009.

[25] M. P. Barcellos, L. P. Gaspary, W. L. da Costa Cordeiro, and R. S. Antunes, "A conservative strategy to protect P2P file sharing systems from pollution attacks," *Concurrency Computation Practice and Experience*, vol. 23, no. 1, pp. 117–141, 2011.

[26] K. Shin and D. S. Reeves, "Winnowing: protecting P2P systems against pollution through cooperative index filtering," *Journal of Network and Computer Applications*, vol. 35, no. 1, pp. 72–84, 2012.

[27] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 2006.

[28] P. Corbett, B. English, A. Goel et al., "Row-diagonal parity for double disk failure correction," in *Proceedings of the 3rd USENIX Conference on File and Storage Technologies (FAST '04)*, pp. 1–14, San Francisco, Calif, USA, 2004.

[29] L. Xu and J. Brack, "X-code: MDS array codes with optimal encoding," *IEEE Transactions on Information Theory*, vol. 45, no. 1, pp. 272–276, 1999.

[30] L. Xu, V. Bohossian, J. Bruck, and D. G. Wagner, "Low-density MDS codes and factors of complete graphs," *IEEE Transactions on Information Theory*, vol. 45, no. 6, pp. 1817–1826, 1999.

[31] M. Blaum, J. Bruck, and A. Vardy, "MDS array codes with independent parity symbols," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 529–542, 1996.

[32] C. S. Tau and T. I. Wang, "Parity placement schemes to facilitate recovery from triple column disk failure in disk array systems," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 89, no. 2, pp. 583–591, 2006.

[33] C. S. Tau and T. I. Wang, "Independent row-oblique parity for double disk failure correction," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 89, no. 2, pp. 592–599, 2006.

[34] B. Cohen, "The bittorrent protocol specification version 11031," 2008.

[35] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos, "File-sharing in the internet: a characterization of P2P traffic in the backbone," Tech. Rep., University of California, Riverside, Calif, USA, 2003.

[36] Y. Kulbak and D. Bickson, "The eMule Protocol Specification," eMule Project, 2005, http://sourceforge.net/.

[37] E. Costa-Montenegro, J. C. Burguillo-Rial, F. Gil-Castieira, and F. J. Gonzlez-Castao, "Implementation and analysis of the BitTorrent protocol with a multi-agent model," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 368–383, 2011.

[38] N. F. Johnson, Z. Duric, S. Jajodia, and N. Memon, "Information hiding: steganography and watermarking-attacks and countermeasures," *Journal of Electronic Imaging*, vol. 10, no. 3, pp. 825–826, 2001.

[39] C. C. Chang, T. D. Kieu, and W. C. Wu, "A lossless data embedding technique by joint neighboring coding," *Pattern Recognition*, vol. 42, no. 7, pp. 1597–1603, 2009.

[40] E. Koutrouli and A. Tsalgatidou, "Taxonomy of attacks and defense mechanisms in P2P reputation systems—lessons for reputation system designers," *Computer Science Review*, vol. 6, no. 2-3, pp. 47–70, 2012.

[41] H. Zhao, X. Yang, and X. Li, "An incentive mechanism to reinforce truthful reports in reputation systems," *Journal of Network and Computer Applications*, vol. 35, no. 3, pp. 951–961, 2012.