

# Towards Access Control Aware Data Access in P2P Data Management Systems

[Draft]

Rammohan Narendula\*  
rammohan.narendula@epfl.ch

Zoltan Miklos  
zoltan@epfl.ch

Karl Aberer  
karl@epfl.ch

EPFL  
Lausanne, Switzerland

## ABSTRACT

### 1. INTRODUCTION

The "Peer-to-Peer" communication model has introduced a significant paradigm shift in the way users share the resources with each other and communicate among themselves. Structured P2P systems based on Distributed Hash Table (DHT) paradigm are increasingly adopted in building massively scalable and highly available data management systems [1]. The academic research community has pursued a wide range of such systems which includes- Chord [2], Pastry [3], P-Grid [4], and CAN [5]. DHT based scalable storage infrastructures are embraced beyond the realms of academia also [6, 7, 8].

Because of the promising characteristics such as massive scalability, zero administration overhead, low cost, and autonomous control, the structured P2P systems are ideal for realizing collaborative networked systems built from resources shared and owned by a cooperative groups of users. Cooperative File System (CFS) [9] reports an efficient robust file system realized with a completely DHT- based decentralized architecture. CFS achieves the performance of traditional FTP- server based file systems, and promises a high robustness to node failures and high availability of the data using replication. A cooperative backup system is proposed in [10] where set of hosts form a cooperative P2P network and backup the data of each other. Each participating computer is assigned to a small set of geographically distributed computers which share their storage to backup the data of the computer. Thus they achieve a cooperative backup scheme which is orders of magnitude cheaper over traditional commercial tape based backup schemes. FARSITE [11] aims at a serverless, secure, and scalable file system targeted for working environments involving few thousand hosts typically a corporate working environment. Efforts are made to build distributed semantic storage infrastructures on top

\*Corresponding author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WOODSTOCK '97 El Paso, Texas USA

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

of P2P cooperative storage. The NEPOMUK project [12] uses P2P semantic storage system and extends the personal computer into a collaborative environment and improves the state of art in online collaboration and personal data management. The TEAM project [13] creates a collaborative P2P semantic infrastructure for lightweight knowledge sharing optimized for distributed software teams. GridVine [14] system builds a P2P semantic RDF store which handles the semantic heterogeneity using schema mappings inserted by peers. A query on the network automatically gets reformulated using these mappings. There are number of P2P based RDF triple stores proposed in the literature [15, 16].

However, for such systems to be widely adopted, there is an obvious need for securing the storage and communication infrastructures. Security, privacy are very critical requirements for any collaborative knowledge sharing system, which motivate more and more users to participate in the system. Any secured system is incomplete had it failed to consider the access Control aware searches and data accesses as its inherent design goals. P2P collaborative systems make such task challenging given that they lack any centralized control and they are realized on top of untrusted resources or nodes. We identify the following concrete goals which must be realized to pursue such a system:

- *Securing the communication infrastructure-* Only intended participants should be desirably, part of the collaborative system. The communication messages exchanged should be visible only to such participants and the communication infrastructure should be free from eavesdropping.
- *Securing the access to the resources shared-* Out of all the intended participants of the network, only certain authorized users should be able to access a particular shared resource. The access control in a system is generally configured using some kind of Access Control Lists [17]. A secure system must support access control aware search and data accesses.

In this paper, we deal with the second part of the security requirements described above. For the first part, we present a brief overview of a simple solution for the P2P secured communication infrastructure that is implemented in the P-Grid project [18, 19]. Later, we demonstrate an extendible modular design for the envisioned Access Control aware P2P data management system (referred as *ACP<sub>peer</sub>*, for convenience, in the rest of the paper).

The remaining part of the paper is organized as follows.  
XXX

## 2. RELATED WORK

The problem of controlling accesses on shared data is widely studied by the research community and researchers proposed a number of access control models and techniques to enable access of the files or resources by only intended users. Most of the existing operating systems implement Discretionary Access Control (DAC) [20] where the owners of the files specify the access privileges for the resources they publish. The management of access privileges is completely at the discretion of the owner only. An arbitrary user can be the owner for the data which he accessed from elsewhere and start sharing this copy with others by managing access rights for this copy. Thus, in DAC-based systems, data can leak to users or processes unintended by the original owner, which can not be restricted. Mandatory Access Control [20] addresses this problem by introducing privilege levels to users and resources. The system monitors the information flow between entities with various privileges and prevents an entity at lower privileges accessing resources configured with higher privileges. Role-based Access Control (RBAC) [20] provides scalable alternative for Access Control Lists, where privileges are configured for a role, users assume. There is no need for configuring the privileges for each and every user of the system.

Granularity of access control is inherently dependant on the kind of access control model deployed in the system. In RBAC, the policy specification granularity can not be finer than the level of roles. Assigning privileges to a single user is not possible unless a special unique role is created for user alone. However, this is not the case with DAC, since it allows granularity to per-user level. If the storage system does not allow flexible handling of such granularity, it leads to data leakage as observed in [21].

Any ACPeer system can be assumed to be analogous to a widely-studied problem of hosting files or data on untrusted storage since data or index is stored often on unknown (and hence untrusted) peers. Such systems use cryptographic techniques to realize access control aware data accesses. The Plutus [22] describes a file system that aggregates files with similar privileges into groups and encrypts all the files in the group with a single key. The key management and distribution is in user's hands and is done out-of-band. SiRiUS [23] describes a filesystem which provides access control in the untrusted storage model using public key cryptography for all metadata operations. Similar public key cryptography technique was used in FARSITE too [11]. SUNDR [24] presents a network file system to store data securely on untrusted servers. Any violation of access privileges or unauthorized modifications by malicious users can be detected by the owners. The work was primarily motivated by to design safe code repository hosting servers of few open-source and commercial projects as traditional centralized storage were observed to be vulnerable to malicious attacks. Most of these works target at securely outsourcing storage to third party storage providers. In [21], the authors address this problem in detail and introduce a robust access control framework using cryptographic techniques. They improve on the existing access control model for UNIX-like systems and apply it to the accesses on data hosted on untrusted storage providers.

However, to be discussed later, the data access patterns in structured P2P systems might vary from that of the above systems. These P2P systems publish certain metadata called index, for each resource shared. Peers first consult this index before accessing the original resources. There is very little work done in the literature on access control mechanisms for structured P2P systems, whose presence is increasingly expanding into a number of different application domains. The work in [25] provides a policy based access control framework for P2P grid systems. An access control system for collaborative environments involving mobile and P2P systems is addressed in [26]. Access Control satisfiability is used in [27] to compute the trustworthiness of acquaintances in a P2P overlay network. Modeling access control in the case of P2P collaborative systems is addressed in [28]. The authors propose a fine granular and attribute based access control framework where each peer assumes a group role and an application role. Then an access control policy which maps various roles and permissions is configured and the underlying framework executes the policy. Our work is based on the lines of PHera [29] which proposes a scalable and fine-grained access control framework for P2P infrastructures. They deal with super peer based P2P overlays where, sub peers specify their access control policy and the super peer enforces it on their behalf. Any invalid request for a resource will not cross the super peers and reach the peers. Policy statements of individual policies are grouped for scalability and performance reasons. However, this work assumes that all super peers are trustworthy to enforce the access control policy of the sub peers. In any case, the sub peers, before processing an access request, can still verify the access control decisions of super peers.

It may appear vaguely that ACPeer is a variant of a P2P anonymous data sharing systems, a problem well addressed in the literature. We contrast both types of systems in the following and argue that solution for the latter can not meet the requirements of the former.

### 2.1 Access Control vs Anonymity

Systems that allow anonymous sharing and publishing of data are of vital importance to protect the identity of the sender and the receiver. Anonymity in such systems refer to *sender anonymity*- hiding the identity of the sender, *receiver anonymity*- hiding the identity of the final destination of a resource request (in DHT based P2P systems, it is the node responsible for the keyword in the request query), and the *storage anonymity*- hiding the actual location of data which might be different from the receiver anonymity. Its unarguable that P2P overlay technologies emerged as the only alternative for realizing anonymity-preserving data publishing systems, which do not need any centralized resources for their functioning, thus eliminating a major source of anonymity compromise. There exists a number of studies devoted to anonymous systems in the literature [30, 31, 32, 33, 34, 35]. The anonymous P2P systems are introduced to stop censorship and filtering of a particular content to make it unavailable for the world. Freenet [30] is a very popular unstructured anonymous P2P system and the Gnutella [36], Kazaa [37] also provide anonymity through the use of the random overlay topology and flooding based routing protocol. But such systems suffer from lack of guaranteed look up, and Agyaat [31] brings the anonymity to the structured P2P systems by creating clouds on top of a P2P overlay to

hide the sender and the receiver identities.

However, even though both the anonymous P2P systems and the ACPeer systems aim at preserving privacy of the data against unintended accesses, both have separate design goals and solution space. Here, we try to highlight the major differences.

1. In anonymous P2P systems, the *access* to a resource is *open*. Anybody can access a resource and the information related to who is initiating the request, who is holding the resource is hidden. In such systems, we can not *control* the accesses to resources, which is the main objective of an ACPeer system.
2. Systems preserving sender's anonymity protect the identity of the sender, but in general, any access control systems needs to know the identity of the requester to evaluate the Access Control Function, which is formally defined in Section 3.5. Hence, systems targeting at only requester anonymity can not meet our objective.
3. In the case of systems providing only the receiver's anonymity, ACF can be enforced by the receiving peers. But we need a secure communication infrastructure and faithful ACF execution by the peers, which may not be the design goals of such systems. The additional protocol overheads introduced in the process of anonymizing the receiver peers is unintended for ACPeer systems.
4. It is clear that systems providing both sender and receiver anonymity can not meet the objective because of the above reasons. Systems preserving storage anonymity alone, also fail to replace ACPeer systems for the same reasons.

## 2.2 Secured P2P Communication Infrastructure

Any P2P system which boasts of secured access control enforcement should first have a secured communication infrastructure in place. There exist widely adopted standards such as Secured Socket Layer (SSL) [38] for secured point-to-point communication. SSL and its updated version Transport Layer Security (TLS) [39] are very well established standards for secured communications over unreliable channels. SSL claims to address every aspect of secured communication—namely *authentication*—peers on either side of a communication channel should authenticate each other, *encryption*—all the messages should be encrypted at the sender and decrypted at the receiver, no unauthorized person should be able to listen to the messages, *message integrity*—the receiving peers should be able to detect any tampering of the messages transmitted by malicious eavesdroppers. A simple SSL-based communication infrastructure was implemented for the structured P2P system P-Grid. One notable instance of SSL based P2P implementation is JXTA [41], which is a widely popular open source project that defines a set of protocols for adhoc P2P computing. SSL encryption is increasingly being used in P2P networks to deliver large files, video over the Internet [40]. The volume of the encrypted P2P traffic has risen ten-fold in just a year and represents more than half of all P2P traffic.

User identities or credentials in the form of digital certificates form the base of security in SSL. The management

of certificates in a decentralized setting is tricky and challenging. One trivial solution could be using self-signed certificates. JXTA uses self-signed certificates, thus eliminating the need for a central control. In collaborative environments coming up with a simple decentralized hierarchical scheme where each group of peers have a single authority to sign their certificates, is an easy task. This reduces the sizes of the trust stores the peer implementations should keep track of and eases the management of such stores. However, developing a scalable, decentralized certificate management is beyond scope of this paper.

## 3. OVERVIEW OF THE PROBLEM

In this section, we present the problem statement along with the necessary background. Then we sketch the modeling of various aspects of the system along with the modeling of the access control policy and we highlight our approach to the problem briefly.

### 3.1 System Description

The goal of the paper is build a ACPeer system using a structured P2P storage. Each structured P2P system is characterized in general, by two primitives—namely **put(key, value)** and **get(key)** where the former is used to publish a data object into the network, where as the latter is used to retrieve the object. **key** is a key to uniquely locate a resource item in the network, and the **value** is either the physical location of the data object or data object itself. Users/applications use the system to publish and search resources/data objects (files, RDF triples data, for example). The exact system architecture and functioning is described later.

One or more users access the ACPeer system using the application built on top. The system consists of several peers (or nodes) which contribute to the storage and processing needs. Thus, in general, either one or more users directly map to a peer which is not configured apriori. However, for ease of description, we assume that a single user in the application domain maps to a single peer in the P2P network, and hence the word user, peer, node refer to the same. However, we claim that the mechanisms discussed do not have any inherent dependency on the user to peer mapping.

In an ACPeer systems, users publish resources and associate access control rules to the data objects which determine who in the network is authorized to access the resource. The exact semantics and the modeling of the policy rules is formally done later in the paper. Each resource has a owner user who is the user first inserting the resource with a certain key, into the network. Other authorized peers can search and retrieve the resource using above primitive. As mentioned earlier, the system adopts a DAC model which does not deal with what happens after an authorized peer retrieves the resource. For example, the system does not prevent an authorized user retrieving a resource from the network and publish it again with different identifier, thus becoming an owner. However, it is to be noted that resources retrieved by peers are not automatically available in the system unless they are published explicitly, thus ensuring that only the original owner has full control on the resource.

### 3.2 What is an ACPeer system?

As noted earlier, an ACPeer system ensures that published data objects are accessed only by legitimate users. Such a

system is formally defined as follows:

In an ACPeer system, there exists, for each published resource  $r$ ,

1. a set of data, to be protected from illegitimate accesses called  $ACData(r)$
2. an access control policy rule  $P$ , which maps  $r$  to a set of peers that can access the  $ACData$ , which is more formalized in later part of the paper

If  $Data(u)$  is all the data that a user/peer  $u$  can access, then ACPeer system must ensure that,

$$ACData(r) \subseteq Data(u)$$

if and only if  $u \in P(r)$ , for any resource  $r$ .

Each resource has two type of data entities to be protected- the resource itself and the *Index* which is stored in the network in the form of  $(key, value)$  pairs discussed above.  $ACData(r)$  is a subset of the set of data entities for resource  $r$ . The ACPeer systems can be categorized based on the members of  $ACData$  set, as explained in the following.

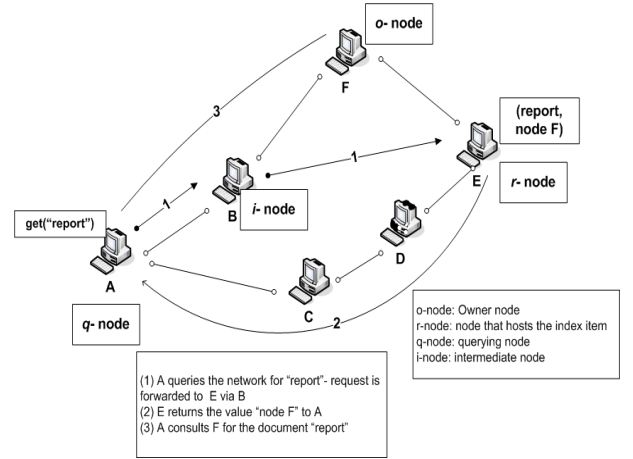
### 3.3 Levels of Access Control

What constitutes the set  $ACData$  for a resource determines the level of access control the system can provide. Based on various requirements of the applications built on top of ACPeer system, we can categorize an ACPeer system as

1. **Level-0 system** if  $ACData(r) = \emptyset$ , for each resource  $r$  published in the system. This system does not provide any access control on the resources. Such a system is suitable for applications which provide raw P2P storage. The conventional P2P systems- Chord, Pastry, etc. are examples of Level-0 systems.
2. **Level-1 system** if  $ACData(r) = \{r\}$ , for each resource  $r$  published in the system. Such a system only protects the resource from illegitimate accesses. Such a system is useful, for example, to host document publishing systems, where every document can be accessed only by authorized paid members, but the index of the documents is world-readable. So any user can search with in the index. For examples systems like *IEEEEXPlore* are Level-1 systems.
3. **Level-2 system** if  $ACData(r) = \{Index(r), r\}$  where  $Index(r)$  represents the index of resource  $r$ . This index may include in addition to the  $(key, value)$  pairs, any sophisticated metadata like descriptors, keywords of the data object. Level-2 systems provide the highest control on the published resources. These systems realize access control aware searches [46] on the index.

In addition to the above classification, one can imagine a system where  $ACData$  covers  $Index$  only partially, where in, access to the value field of the  $(key, value)$  pair is controlled but the key field is not. Such systems can be labeled as **Level-1.5 system** in this context. Such systems allow any user in the system to figure out whether a resource with particular identifier exists in the system, without revealing anything more about the resource. To be seen later, a Level-2 system entails a non-negligible overhead in terms of search and publication cost over a Level-1.5 system.

Here, we try to scrutinize the transaction flow in a typical structured P2P system and illustrate the levels of ACPeer system with examples. Figure. 1 illustrates the typical work



**Figure 1: A typical transaction flow in a structured P2P system.**

flow of a structured P2P system. A peer  $F$  wishes to publish a resource with name *report* in the network. This node is termed as the *owner node* (*o-node*) in the model. The underlying routing algorithm decides to host the index information about this resource ( $Index(report)$ ), say a pair  $(report, F)$  on node  $E$  which is termed as the *responsible node* (*r-node*). Any peer can query for this resource using the *get-* primitive. In the figure, the querying peer (*q-node*)  $A$  queries for this item and the query is forwarded to node  $E$  via several *intermediate nodes* (*i-node*).

Assume that the owner node gives only node  $A$  the required access rights for this resource.

- In Level-0 system, since access control is not in place, node  $A$  and a node like  $C$  which is not intended by the owner, can query the system to get the location of the owner node, and then retrieve the resource.
- In Level-1 system, both nodes  $A$  and  $C$  figure out that node  $F$  hosts the resource *report*.
- In Level-1.5 system, node  $A$  can only figure out that *report* is hosted on node  $F$ , but node  $C$  will get reply for query on *report* in such a way that it can infer that a resource with identifier *report* is published in the system, but it can not make out about where it is hosted.
- In Level-2 system, only node  $A$  can successfully know where it is hosted. Node  $C$  can not even know whether a resource with such identifier is published in the system or not.

### 3.4 Problem Statement

Given a structured P2P system, realize a Level-2 like ACPeer system.

### 3.5 Modeling Access Control Policy

A typical access control policy categorizes the incoming access requests in terms of the principals identified in the

system such as user identifier or group identifier, and maps each available resource to one or more of such principals which can access the data. For instance, access requests can be categorized based on the source or the user where the requests are coming from, where the users or groups of users form the principals. The categorization criterion determines the list of principals available in the system. Categorization can happen based on the roles the requesters assume in the system. Another way of categorization is kind of relation the requester is having with the publisher of the resource, on the social network. Hence, any access control policy at a peer which defines access privileges for each resource in terms of the set of principals available in the system. In DAC model, it can be represented by an Access Control Function (ACF) of a peer  $p$  which can be formalized as follows:

$$ACF^p : \aleph \rightarrow \bigcup_{\forall r \in \mathfrak{R}^p} ACData(r)$$

where  $\aleph$  is set of principals and  $\mathfrak{R}^p$  is the set of resources published by  $p$ . The set  $\aleph$  grows exponentially w.r.t the number of peers in the system,  $\aleph$  is in the order of  $2^{|\Theta|}$  where  $\Theta$  is the set of peers.

The above policy modeling assumes only read access privileges for the principals. In general, the policy specification should consider the set of allowed operations on the resources published. To be explained further, the paper does not address the issue of principals management in the network. In section 3.6, we discuss the assumptions we make in related to access control policy specification. In section 3.8, we present a general approach to the given problem which does not make assumption of a particular model of the principals.

### 3.6 Assumptions

In order to exercise greater flexibility to concentrate on the original problem of designing ACPeer system, we resort to few simple assumptions, which are enumerated in the following.

1. Unless specified explicitly, we assume the storage model where the resource is stored with the  $o$  – nodes itself, and the *value* part of (*key, value*) points to the location of the  $o$  – node. The solutions are adoptable to other storage models where resource is stored inside the network on the  $r$  – nodes for example.
2. Since establishing identity of a principal securely is a critical prerequisite for any access control system, we assume the availability of the same. We assume existence of a secured function which maps an access request to one of the principals available in the system, using which the reference monitor can either allow or deny the access request based on the policy statement. We assume the existence of PKI infrastructure which includes certificate authorities (CAs) who can sign identity and public key certificates of users, a key distribution mechanism. Some of the approaches depend on this infrastructure for all aspects of the system functioning- authentication, publishing, searching, where as some solutions use the infrastructure only for authentication of incoming access requests.
3. We assume that in a typical knowledge sharing system, there are many reads but rare updates or writes on the

same data. Hence we concentrate only on controlling the read accesses on the data published and leave controlled write accesses as a future study. However, with little enhancements, the solution mechanisms can be fine tuned to write accesses also.

4. We assume a simplistic threat model where peers try to abuse the system by trying to access the data they are not authorized to access. Otherwise, they cooperate with other aspects of the routing protocol- for instance, we do not assume that peers delete the data they have to host as per the protocol, and peers do not abuse with the routing algorithm of the P2P infrastructure. This simplistic model relieves us from several complicated issues and enables us to concentrate on only the access control aspects of the data accesses. However, we recognize that mitigating every possible abuse on the system is beyond the scope of the paper.

### 3.7 Motivation

Before proceeding to realization of Level-2 system, here we provide the motivation behind envisioning such a system.

- Imagine a P2P information system inside a corporate company- managers, employees, and temporarily hired contract staff share the resources over the P2P platform. In open P2P systems (like Level-0 or Level-1 systems), even the contractors can place queries for say appraisal review related docs and infer something like whether appraisal of some employee is ready or not or is it modified after the employee met the manager, if this kind of metadata is included in the *Index* of the resources. Typically managers may want only other managers or some employees whom he chooses, to see what he is sharing in the network, in spite of the fact that the actual access to resources is restricted. Imagine a knowledge sharing system. Adversaries can query the system with few keywords and infer what kind of knowledge a peer is contributing to the system- say *whether any docs in the system exist about virus xyz?*. A Level-0 or Level-1 system can easily expose such data through uncontrolled index accesses on the metadata or index. The authors in [21] demonstrate the threats due to such data leakage in traditional file systems.
- In a Level-2 system, a query from an ineligible peer is stopped as early as possible from a typical successful query-reply cycle which saves on the communication costs. Such a peer does not get any useful reply packets for a query term, it can not proceed further on the transaction flow.
- In case of very popular resources, usually the load on the serving peer is high. If the queue size on such peer grows beyond an acceptable limit, all future requests may be denied which might include requests from eligible peers. So a Level-2 system like ACF enforcement tries to fill the queue with only valid peers.
- In systems like GridVine, a single query translates into multiple queries recursively, when matching semantic mappings or translations are found in the network. In Level-2 like systems, we can prevent queries from

unauthorized peers from going deep into the network recursively.

- More importantly, in some systems, there may not be any physical resource associated with the index to enforce access control at the end nodes. In this case  $r = \emptyset$  and  $ACData(r) = Index(r)$ . The resource itself would be stored in the network e.g., in the form of an RDF tuple. In such case, the only available point of ACF enforcement is the network itself. Level-0 or Level-1 systems do not control accesses in such cases. In the TEAM project [13], peers share knowledge which is in the form of instances of OWL ontologies, each knowledge unit is published into the network and does not have any associated physical resource on the publisher node. Such systems need Level-2 AcPeer systems.
- In second and third types of storage models, the resource does not reside with the publisher. Hence, a Level-2 system is required for such storage models where the same technique used for enforcing ACF on index can be applied on the resource also. More over, its common in data sharing systems to split a published resource into fragments and distributing the fragments in the network, for enabling faster parallel downloading of popular big files. In this case, some fragments will be accessed without the knowledge of the owner, which demand a Level-2 system like ACF enforcement.
- An adversary may query the system for particular keywords and launch DoS attacks on the serving peers using the information he/she gets in the replies.
- If index is open, an adversary can construct the history of who are all searching for the key it is hosting from the queries it receives- and this valuable history may be used to launch some other kind of attacks. In Level-2 systems, this is not possible, because a random adversary can not know about the index just because it hosts it, unless it has access to it.
- As explained earlier, sophisticated indexing mechanism can be built on top of the basic indexing primitives provided in the form of key,value pairs. An inverted index of the documents published, is one of such possibilities. Any indexed document can be constructed with ease from the open inverted indexes of the document [44]. P2P information retrieval systems based on distributed inverted indexes are pursued in the literature [45]. Hence, a Level-2 like AcPeer system would be a first step towards access controlled P2P information retrieval systems.

### 3.8 Our Approach

The goal of the paper is to enforce the ACF of the publishing peers so as to meet the requirements of the Level-2 system. In the literature, entities which enforce the ACF are referred as *reference monitors* [20]. From Figure. 1, it is clear that a Level-2 system requires to realize reference monitors on nodes other than the owner. Hence, in the case of P2P AcPeer systems, there exist two points to realize reference monitor i.e., *o-node* and *r-node*. Implementing a trusted reference monitors in the latter case is trivial as it is done locally. However, realizing reference monitors on former (i.e., o-nodes) is non-trivial and can be done in several ways as demonstrated later in the paper.

As presented earlier, any AcPeer system has mainly two concerns- first, modeling an access control policy- selecting the set of principals present in the system, and second, implementing a trusted reference monitor. To address the first concern, in stead of restricting ourselves to a specific type of principal modeling (such as role based modelling), we present a generic approach which assumes a general principal and sketch a modular design for the targetted Level-2 system, where, we provide a general framework for the second part of the problem- implementing trusted reference monitor, into which a module which solely deals with a specific type of principals is plugged in, in order to realize a complete functional Level-2 system to be used by applications. For example, if we pursue an encryption based mechanism for implementing a reference monitor, we assume encrypting the resources per user if the policy models principals in terms of users, or encryption per role if a model based on roles is pursued. For illustration purposes, we assume a conventional modeling of principals- users and groups where each peer is assumed to be a user. Groups of peers can form a single principal.

The most critical and challenging objective of any AcPeer system is to guarantee a faithful execution of ACF. It must be noted that, on a Level-2 system, ACF must be enforced on  $Index(r)$  which is essentially hosted on peers other than the publisher itself. The paper is mostly devoted to this problem of implementing trusted reference monitors on remote peers keeping the Level-2 system requirements. Some of the solution approaches to be discussed in the paper make an attempt to realize first a Level-1.5 system which would be extended, then, to Level-2 system.

## 4. SOLUTION MECHANISM

In this section, we explore the problem of designing a Level-2 AcPeer system in detail, introduce the following mechanisms to solve the problem, and provide a comparison of all the mechanisms.

1. Disjoint P2P Networks (DPN)
2. Controlled Queries (CQ)
3. Controlled Replies (CR)
4. Hybrid Solution

The DPN approach address the problem by constructing disjoint networks depending on certain objective as explained in next section. The CQ approach realizes the reference monitors in the network using encryption techniques where as CR, in an encryption- free manner. The hybrid solution attempts to inherit properties of both the CQ and CR approaches. These solutions are discussed in detail in the following sub sections.

### 4.1 Disjoint P2P Networks (DPN)

A simple and trivial way of realizing a Level-2 system is by creating multiple independent P2P networks each with its own resources published and the peers participating. A new network is constructed based on, which one of the search cost or publication cost has to be optimized. If publication cost has to be optimized where each peer publishes each of its resources at most once irrespective of number of principals authorized to access, a network with only peers corresponding to authorized principals is created. Resource is published

---

**Algorithm 1** Simplified algorithms for DPN

---

{The algorithms are simplified for brevity. Each step of algorithm may involve additional actions than mentioned- for example, creating a network involves informing the members and issuing membership certificates. Each algorithm is run at each peer  $p$  in the system.}

*Publish\_MinPubCost()*  
{Algorithm for the case of the optimizing publication cost}

**for all** each resource  $r \in \mathbb{R}^p$  **do**  
  create a network with principals  $n \in ((ACF^p)^{-1})(r)$   
  publish  $ACData(r)$  into the network  
**end for**

*Search\_MinPubCost()*  
{Algorithm for the case of the optimizing publication cost}

**for all** every member network **do**  
  search for the item in the network  
**end for**

*Publish\_MinSearchCost()*  
{Algorithm for the case of the optimizing search cost}

**for all** resource  $r \in \mathbb{R}^p$  **do**  
  **for all** principal  $n \in ((ACF^p)^{-1})(r)$  **do**  
    publish  $ACData(r)$  into the network corresponding to the principal  
  **end for**  
**end for**

*Search\_MinSearchCost()*  
{Algorithm for the case of the optimizing search cost}

**for all** member principal **do**  
  search in corresponding network  
**end for**

---

into this network so that it is visible only the  $o$ -node and the member principals. In this case, a  $q$ -node has to search in each of the network it participates being part of one of the member principals. Each  $o$ -node has full control on the members of a network and can use simple admission control protocol where the  $o$ -node issues a digitally signed certificate to each valid member peer.

A DPN approach which aims at optimizing the search cost, will one network per each principal so that search by a  $q$ -node is limited by the number of principals it is member of. However, the publication cost here is equal to the number of principals that are authorized to access a resource. Each principal will control the membership of the network in the same way as explained above. In case of principals with a group of peers, one peer can be elected as administrator which controls the issuance of the membership certificates.

This way DPN ensures that index is available only to the authorized peers, by modeling the access control problem as an admission control problem. Access control is exercised at the time of joining the network itself. The problem of admission control in P2P systems is well addressed [47, 48], which is not discussed here.

However, DPN approach has inherent scalability and efficiency problems, as the number of networks needed would explode exponentially. For the first case, the number of networks would be in the order  $O(|\Theta| 2^{|\mathbb{N}|})$ . In the latter case, it would be in the order of  $O(|\Theta| |\mathbb{N}|)$ . The network and the participants in a network become highly unmanageable as the number of resources and peers increase. However, the

advantage with this approach is, one can exercise the finest granular access control- right to the level of per principal-resource access constraints. This is useful for applications that can not tolerate even minimum amount of information leakage to unauthorized peers.

The DPN solution tries to attack the problem of Level-2 ACPeer system, by modeling the access control around  $o$ -nodes (refer Figure. 1) only and hence end up in non-scalable systems. By moving parts of ACF enforcement to entities responsible for respective phases of a transaction (as described in Figure. 1), we can design better solutions w.r.t both scalability and manageability, which will be elaborated further in the following sections. Such an approach results in primarily two alternatives which realize the Level-2 system by executing ACF at either  $q$ -node itself or at  $r$ -node, which are termed as *Controlled Queries* and *Controlled Replies* approaches respectively.

## 4.2 Controlled Queries (CQ)

The intuition behind the CQ based approach is to embed ACF into the index generation process itself by encrypting the resource identifier space. By ensuring that only the authorized principals get access to the keys, the  $o$ -nodes can ensure proper enforcement of the ACF. Such encrypted items can be hosted safely on any peer in the network. Only the peers with valid encryption keys can pose queries for the items and decrypt the results received. There is no need of any ACF execution on  $r$ -nodes in the network, as it is already enforced at query formation itself, thus completing the ACF execution on index (at  $q$ -node itself). Ability to generate a query with valid query terms itself is considered as the necessary authorization required for the access. This way of ACF execution on the index, backed by the ACF execution on the resources at the  $o$ -nodes completes the realization of a Level-2 system. Name obfuscation schemes were used before, for censorship-resistant publishing in P2P systems [30]. The most promising advantage of a CQ based solution is- there is it does not need any changes to the routing or indexing protocol of the underlying P2P system. An access-controlled query works like any other query and processing the requests is quite transparent to the peers and the network. It can be applied on top of any existing structured P2P system.

Based on the number of keys to be distributed, the CQ based approach can be used to realize either a Level-1.5 or Level-2 system. In the following, we first introduce a Level-1.5 CQ based ACPeer system which will be tuned to a Level-2 system later with additional encryption keys and increased search cost.

Level-1.5 ACPeer System- Publishing and Searching: One simple way of realizing CQ based Level-1.5 system is the following which uses public key cryptography: each principal (a user or a group of users) in the system (from the set of principals  $\mathbb{N}$ ) will have its own trusted pair of public, private keys. The index is inserted into the network after encrypting it with the public key of all the authorized principals as dictated by the ACF of the peer. So a peer can search index published to a principal if it knows the respective public key and can interpret the results *only* if it has the respective private key. This simple solution suffers from dictionary attacks on the index thus compromising the semantics of a Level-1.5 system, which is explained in more detail in later part of the subsection.

---

**Algorithm 2** Simplified algorithms for CQ

---

```
1: {An instance of the algorithms is run at each peer  $p$  in
   the system.}
2: Publish()
3: for all resource  $r \in \mathcal{R}^p$  do
4:   for all principal  $n \in ((ACF^p)^{-1})(r)$  do
5:     publish  $ACData(r)$  into the network
6:     {This step translates to Publish_Level1.5() or Pub-
7:     lish_Level2() depending on which system is de-
8:     ployed. }
9:   end for
10: end for
11: Search()
12: for all member principal do
13:   search for the item with principal's credentials
14:   {This step translates to Search_Level1.5() or
15:   Search_Level2() depending on which system is
16:   deployed. }
17: end for
18: Publish_Level1.5(resource r, principal n)
19: encrypt key of Index(r) with public key of  $n$  using the
20: deterministic encryption algorithm
21: encrypt value of Index(r) with public key of  $n$  using
22: the non-deterministic encryption algorithm
23: publish encrypted (key, value) pair into the network
24: Search_Level1.5(resource r, principal n)
25: encrypt key corresponding to  $r$  with public key of  $n$ 
26: using the deterministic encryption algorithm and send
27: query
28: decrypt the result with private key of  $n$ 
29: Publish_Level2(resource r, principal n)
30: generate a secret key and encrypt (key, value) of
31: Index(r) with this secret key
32: publish encrypted (key, value) pair into the network
33: encrypt the secret key with public key of  $n$  and publish
34: into the network
35: {If the secret key is distributed to  $n$  using out-of-band
36: mechanisms, the above step is not required}
37: Search_Level2(resource r, principal n)
38: for all potential publisher do
39:   {or for each secret key used to publish to  $n$ }
40:   retrieve the corresponding secret key
41:   encrypt key corresponding to  $r$  with the secret key
42:   and send query
43:   decrypt the result
44: end for
```

---

Since encryption has to be done for each authorized principal, an ACData item, must be published as many times as the number of such principals specified in the policy, for the resource. This overhead is in the order of  $O(|\mathcal{N}|)$ , which can not be avoided if encryption techniques are used to realize ACPeer system. Existence of several copies of the same piece of data is greatly disadvantageous when resource updates are made. Similarly, a search for interested index entry must be done separately for each principals for which the peer has credentials (public and private key pairs) for. Hence, number of search queries is also in the order of  $O(|\mathcal{N}|)$ .

Potential Dictionary Attacks: The foremost objective of the index encryption algorithm is to preserve the semantics of the search process in the form of *get()* primitive, hence it must enable searchability of the encrypted index through the *key* part of the index entries. Searchability is achieved only when *q-nodes* can reproduce the same cipher text equivalent to the interested ACData item, as was generated by *o-nodes* during publishing. This is possible only by using deterministic encryption [49] algorithm (such as RSA/NoPadding). A deterministic encryption ensures that a given plain text and key combination always compute to the same cipher text every time the algorithm is applied. The downside of such approach is that the *value* field of index, is subject to dictionary attacks. An arbitrary peer, knowing the public key of a principal, can place a search request on the items accessible to only that principal and then construct the complete index entry using a simple dictionary attack. For instance, an arbitrary peer can know where a resource *virus\_patch.doc* is stored, by posing the query for the key by encrypting with the public key of the authorized principal. He can later try with a list of known peers and encrypting the peer ids with the same public key and comparing which one of them matches to the reply obtained for the search query, thus completing the attack.

However, fortunately, to desist the leakage of the *values* to unintended peers due to dictionary attacks, all the value fields of index entries will be encrypted using a *non-deterministic* cryptographic technique. In a non-deterministic encryption (such as RSA/Padding), a given plain text and key combination compute to a new cipher text every time the algorithm is applied. However, all such ciphers decrypt to the same plain text. This way its impossible to figure out which one of known values maps to the encrypted value as every time the algorithm is applied, the cipher text varies. Only a authorized peer which has the respective private key can decrypt the value field. Note that if non-deterministic technique is used for the *key-* part of index, the searchability is lost because the peers can never reproduce the same cipher originally generated by the publisher.

Level-2 ACPeer System- Publishing and Searching: The above Level-1.5 system can be improved to a Level-2 system by encrypting the ACData items with a secret key (instead of public key of principals) for each principal, which is known only to the publisher and the authorized principal. A publisher generates a secret key and encrypts the item with the secret key, and distributes this secret key either by offline mode or by encrypting it with the principal's public key and publishing it into the network. The authorized peer first retrieves the encrypted secret key and decrypts it with its private key, then queries the system with a cipher text equivalent to the query term encrypted with this secret key. Since the secret key is not available to other unauthorized

peers, they can not frame valid queries. However, this solution has inherent scalability problems as the number of such secret keys required would be in the order of  $O(|\mathcal{N}| \cdot 2^{|\mathcal{N}|})$ . In addition, the search overhead also increases significantly as the number of search queries required to search for an interested item would be in the order  $O(|\mathcal{N}| \cdot |\Theta|)$ , as one peer has to search with each accessible principal and for each potential publisher.

**Revocation:** It is clear that the CQ based approach relies on a secured distribution of principals' credentials to the potential publishers in the system. As a result, the ACPeer system should support revocation of access privileges due to not only the updates in ACF by the publisher peer but also, but also credential updates by some principals. For example, a principal corresponding to a group of users may update its credentials if the group membership is updated. Revocation in CQ-based ACPeer systems can be realized if and only if additional primitive, namely `remove(key)`- to remove an index entry is provided by the underlying DHT. ACData items published with old credentials can be removed from the system and republished with new credentials, when a revocation is requested. This primitive must be realized in a secured way allowing only the original owner to issue this operation on the keys he/she has published. Given the fact that the ACData items are stored on arbitrary untrusted peers, implementing a reliable deletion operation is challenging.

To summarize, the CQ based approach keeps the network protocols intact for realizing a Level-1.5 and Level-2 system, but requires key management and distribution infrastructure. CQ approach proves to be more suitable for a Level-1.5 than a Level-2 system. Knowing that an arbitrary peer which has the public key of a principal can only issue search queries, public keys should not be distributed indiscriminately in the network. In a typical collaborative working environment, exchanging of public keys is normally done in a controlled manner (somewhat similar to distributing email addresses or personal phone numbers). In such cases, an application which need Level-2 semantics can still resort to a Level-1.5 system based on CQ approach to avoid the additional overhead caused by Level-2 system. Restricted distribution of public keys limit the exposure of the key-part of index items to arbitrary peers in the network.

In the next section, we demonstrate the Controlled Replies (CR) approach based Level-2 system which pushes the ACF execution into the realms of the  $r$ -nodes resulting in a very simplified system free from complicated key management inherent to the CQ-based approach.

### 4.3 Controlled Replies (CR)

In this approach, index entries are stored in an encryption-free manner, hence, any peer (even unauthorized peer) can post queries for interested data as the index space is not obfuscated, in contrary to the CQ-based approach where peers having the credentials can only pose queries. CR-based Level-2 system ensures that queries that originate only from the authorized peers are replied i.e., access to index is controlled at  $r$ -nodes in the same way as done for the original resources at  $o$ -nodes. The  $r$ -nodes implement the necessary reference monitor. Part of complete access control policy (ACF) corresponding to the respective *key*, is stored at  $r$ -node along with the index entry, and is checked against for each incoming query for the key. Since ACF is executed

by  $r$ -nodes, we need a trustful mechanism to ensure that a particular  $r$ -node executes the ACF faithfully. To this end, we introduce the technique of *constrained indexing* where  $o$ -nodes can control where the index entries corresponding to the resources they publish, must be stored. The intuition is that  $o$ -nodes can choose such a set of peers based on the social relationships which host the ACData items on each other and implement the reference monitors. In the following sub section, we introduce such a technique which forms sub overlays with only trusted peers on top of the overlay. We present insights to various issues arise as part of the solution and introduce a technique which makes several promising improvements.

#### 4.3.1 CR with Trusted Sub Overlays (TSO)

Since traditional structured P2P systems assume all peers to be the same, every peer can be a potential candidate for storing index of the resources published by other peers. Such *anybody-can-host-anybody's-index* paradigm is no longer suitable for realizing a faithful reference monitors. To this end, we visualize a *constrained indexing* technique where trustworthy communities of peers are formed and members of the community decide together to host on each other, the index of the resources they share into the network. These are the groups of publishers i.e.,  $o$ -nodes. Each such community forms a suboverlay on top of the main overlay which is termed as *Trusted Sub Overlay (TSO)*. Each TSO has its own identifier space (equivalent to that of the main overlay) for the published resources by the TSO members. A peer in this model, plays two roles one as a member of the main overlay and the second as a member of one or more TSOs, hence is responsible for its identifier space as part of the main overlay and that of the affiliated TSOs. Figure. 2 demonstrates the concept of TSO. Peers  $p_1, p_9$ , and  $p_3$  are part of  $TSO_1$  and another overlay  $TSO_2$  has  $p_4, p_6, p_8$ , and  $p_9$  as its members. Node  $p_1$  will publish its data items that need controlled accesses into  $TSO_1$  and any open data items into the large overlay which includes all the peers from  $p_1$  to  $p_{11}$ .

As a result, any resource that is hidden inside a TSO must be identified globally by both the TSO id and the identifier local to the TSO. To make this possible, we augment the traditional flat identifier space used to identify the resources in conventional P2P systems, with a two dimensional identifier space to identify resources that require controlled accesses (hidden inside a TSO)- one dimension for identifying the TSOs and the other dimension for individual resources with in a TSO. Index and resources that can be publicly accessed by any peer are published into the main overlay, which are identified with flat identifiers. Thus, resources which need controlled accesses are identified with a pair of identifiers (`TSO_ID, resource_ID`).

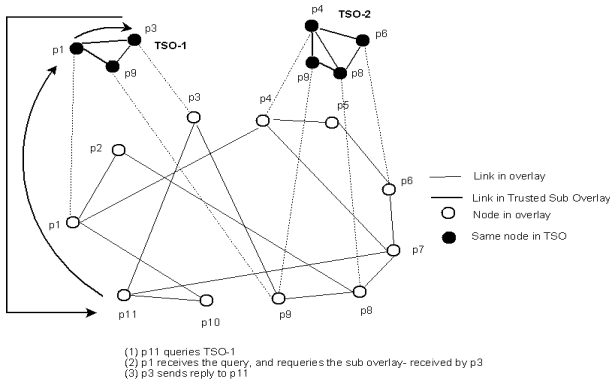
As explained later, the TSOs are used to store ACData items where as the main overlay is used to host any public data including metadata that helps arbitrary peers to discover and query the TSOs. One form of constrained indexing where the resource indexes are localized based on the domain names is presented in [50].

**TSO Formation:** In this model, a set of peers together decide to form a trusted sub overlay, at an arbitrary point of time. A natural choice for such a trusted group of users is all members of a research unit in a huge research organization, or a subset of members of the same project they

are working for. There exist several possibilities regarding how the peers meet in first place to decide on the sub overlay formation. It can be done in a centralized fashion- a single peer (TSO administrator) can decide to start a TSO and invite other trusted members into this TSO, or in a decentralized fashion- where peers in the community use some admission control protocol to extend the TSO. An automatic TSO formation engine can be built which exploits a social network graph of the peers which forms the TSOs automatically based on some configuration settings- like all peers which are directly connected to a peer can be part of all TSOs created by that peer.

Any peer that possess the properties desirable for faithful ACF execution/trusted reference monitor can be a potential candidate to be included in a TSO. It is assumed that a member peer behaves appropriately and cooperates in various aspects of the ACF enforcement including but not limited to, responding to the policy updates from the owner nodes, executing ACF while replying to the incoming query requests, and not compromising the system by any means, for example, sharing the hosted index with unauthorized peers through back door.

Thus, a TSO can begin with one or more peers and extend as new peers join by sharing the same TSO identifier. Thus the new peers share query load and contribute to the storage of the TSO. We assume that the TSO identifiers are generated in a unique way as the resource identifiers, for example, using hash of IP address of the TSO creation node together with time of creation. New peers can join a TSO in two ways: by *invitation-only* mechanism where a member peer invites a non-member peer to join the TSO, by *self-join-request* mechanism where the new peer itself initiates a *TSO-join* request. A TSO's historical record of ACF execution would be an excellent reputation metric which enables a trust worthy TSO to be a favourable TSO to be member of. This requires techniques for quantifying and modeling the trust and reputation of TSOs. Such techniques already exist for the case of a single peer [51] which must be extended to the case of group of peers, which is a research problem in itself. All the ACData items published into a TSO are exposed to all the members of the TSO, which results in the violation of the ACF rules which is explained in more detail in the following.



**Figure 2: Illustration of Trusted SubOverlays.**

Suboverlay Management: An interesting issue that arises in this model is the routing and management in the sub overlays- whether the same overlay technology used for the main over-

lay should be used for the sub overlays or a different overlay technology tailored to a TSO needs should be used. For example, a TSO topology can be modeled as a broadcast topology (like in Gnutella) given the small number of nodes in a TSO, where as the main overlay continues to be a structured overlay. In fact, each TSO can have its own topology and overlay management mechanism independent of all other TSOs in the system. In some prospective, this model is interpreted as the main overlay forming a huge substrate into which several independent secured small substrates are plugged. This way, each TSO can be plugged into the big overlay any time so as to make the resources available to other non-member peers selectively, and can be similarly unplugged from the overlay. To be explained shortly, a TSO is plugged into the overlay by publishing some unsensitive data about the TSO which enables other non member peers to discover the TSO and query for the resources published in the TSO.

Publishing: An ACData item is published into a TSO using TSO- specific primitives for publishing. These primitives may be same as that listed in the paper earlier, if the TSO is managed with a structured overlay technology. A publish request message must be routed through only the members of TSO so as prevent any leakage of the index during publishing phase. Once published, an ACData item is visible to a set of member nodes of the TSO, which makes it possible for the members to view the index if they wish. This set includes the member(s) which is (are) responsible for the identifier space corresponding to the published resource and all the intermediate nodes participated in the *put* operation. It should be noted that such member nodes may not be authorized to access the resources as per the ACF. However, we believe that this *leakage* is tolerable given the kind of peers that are present in a TSO. This leakage for a TSO with identifier  $t$  w.r.t a member peer  $m$  can be quantified as

$$L^m(t) = \{ \sum_{\forall r \in \mathcal{R}^m} | M(t) - (ACF^m)^{-1}(r) | \} - 1^1$$

where  $M(t)$  is the set of member peers (referred as *Member list* or *Friends list* in the rest of the paper) of the TSO  $t$ . Thus, the total amount of *leakage* in a TSO  $L(t)$  can be quantified as

$$L(t) = \sum_{\forall m \in M(t)} L^m(t)$$

It should be noted that  $L(t) = 0$  if and only if either  $M(t) = 1$  or every member of  $M(t)$  can access every resource published by each of the other members. For very sensitive ACData items, which can not tolerate any leakage, the TSO size can be kept to be only one where the items are stored on the owner peer itself. The other alternative is to include only the members in a TSO such that leakage always amounts to zero. The latter way of TSO formation is rather very difficult as it is not always possible to find such a set of peers which have access to every resource published by each other. Hence, it is always the case that there exists some leakage in a TSO. However, we believe that, since a TSO has typically a fewer number of nodes because TSOs are modeled on social relationships which are limited in number, the amount of leakage is always in tolerable limits.

<sup>1</sup>-1' is because  $M(t)$  includes the member  $m$  itself which is not present in ACF.

Members of a TSO must publish certain metadata about the TSO into the main overlay so as to enable other non member peers to discover the points of contact (entry points) for the TSO. An *entry point* of a TSO is any member node of the TSO. The use of entry points is explained when the search process in TSOs is explained later. The number of such entry points in a TSO need not necessarily be equal to number of member nodes of the TSO so as to facilitate some nodes hiding their TSO membership information. Entry point list is a collection of  $(TSO\_ID, node\_ID)$  pairs. The authenticity of such an entry point list for a TSO can be ensured using cryptographic techniques- for example, each entry digitally signed by the administrator of the TSO. We are investigating sophisticated techniques in this regard. There can be certificate authorities issuing entry point list certificates to all legitimate members of a TSO. This way, the authenticity is verified in the same way public keys are verified in PKI infrastructures.

When a peer is about to join a TSO, it can measure the leakage in the TSO w.r.t the peer, and can ignore to join the TSO if such a leakage is higher than a threshold. In an invitation-only mechanism, the inviting peer can hand over the necessary data to compute such leakage metric. We assume all the control activities necessary to include a new members into a TSO are done through out-of-band channels (like emails or other personal communication channels), however providing an in-band mechanism is a goal we pursue as part of future study. One of the main control activities relates to admission control- how the current members of the TSO agree on inviting a new peer and how the status of a successful join of a new member is propagated to all members. A very simple approach could be a *quorum-based admission control* where in at least a minimum quorum of the current group should accept the new member to be part of the TSO.

The friends list of a TSO can be cached locally on every member peer as the TSO grows. Given the size of the TSO, updating caches on all members, when a member joins or leaves a TSO, can be done with a reasonable communication cost. In self-join-request mode, the issue is complicated- how to safely retrieve the list of TSO members so as to compute the leakage metric is challenging given that, not every peer wants to disclose its membership information in a TSO. For the rest of the discussion, we assume that a peer joins by invitation-only mechanism only.

It should be noted that a single peer can be part of multiple TSOs each tailored (w.r.t the membership and topology) to the kind of items hosted inside the TSO. Hence, it can publish its resources into more than one TSO with selectively choosing which set of resources into which TSO, based on several criterion. One of the criteria is the reputation of the TSO in executing ACF. Other criterion is to choose the TSO which minimizes the leakage.

Searching: An arbitrary peer can search for the resources published into the main overlay in the same way using `get()` primitive discussed earlier. However, the search for access controlled data that resides inside TSOs is done in two phases. First at least one entry point of a TSO should be retrieved from the main overlay. This is done by querying the main overlay with the TSO identifier. Then the original query should be passed to this entry point. One entry point is chosen randomly in case there are more than one available. Once the query enters into the TSO overlay, that sub over-

lay's routing mechanism dictates where this query should be forwarded from that entry point. Once the peer responsible for that identifier space sees the query, it invokes the reference monitor which replies to the query. Thus, this simple access control aware search mechanism meets the access control requirements of the Level-2 system we sketched in the beginning of the paper.

However, the main question to be asked here is, how the  $q$ -nodes know about in which TSO to search for? It is clear that with the above simple search mechanism, each peer has to search in each TSO separately, thus the number of search queries is in the order of the number of TSOs in the system. This increased query traffic overhead is acceptable given the very nature of the problem of ACPeer system. However, if the querying peer knows about the TSO it has to search for, there will be maximum two queries over the system- first to retrieve the entry points and then the actual query. However, as an improvement, we envisage sophisticated techniques where TSOs publish *data descriptors* which capture general broad description of the data hosted inside a TSO without revealing the identities of the actual data or index, thus not violating the requirements of a level-2 system. For example, a TSO that hosts a software project related AC-Data, can have all the keywords describing the project in the data descriptor.

---

### Algorithm 3 Simplified algorithms for CR-TSO

---

```

1: {The algorithms are simplified for brevity. Each algorithm is run at each peer  $p$  in the system.}
2: Publish()
3: {The algorithm assumes that  $p$  selects a single TSO for hosting all of its resources.}
4: choose a TSO  $t$  out of all known TSOs such that  $L(t)$  is minimum
5: for all resource  $r \in \mathcal{R}^p$  do
6:   publish  $ACData(r)$  including  $((ACF^p)^{-1})(r)$ , into  $t$  using  $t$ 's primitives for publishing
7:   if  $Level - 1.5$  access control is the requirement for  $r$  then
8:     publish  $(key, TSO\_Id)$  pair into main overlay using put() primitive
9:   end if
10: end for
11: Optionally, join entry point list of the TSO
12: Search()
13: if The TSO to search in, is already decided then
14:   {either using data descriptors OR interpreting the TSO identifier OR from the list of TSO identifiers retrieved by querying main overlay for interested key}
15:   retrieve entry point list of the TSO
16:   send query message to one entry point for the interested key
17: end if
18: if the TSO to search in, is not decided then
19:   repeat above in each TSO known
20: end if

```

---

For the index which does not require Level-2 access control,  $(key, TSO\_ID)$  pairs can be published into the main overlay for all the index entries so that all the TSOs where a key is available can be retrieved with a single query. Hence, peers will not end up in failed searches in TSOs that do not host a particular key. This improvement provides Level-1.5

semantics for ACPeer system.

**Mitigating updates in ACF and TSO membership:** One very promising advantage of CR-TSO approach compared to other schemes discussed earlier is the ease to handle the updates in ACF: addition of new rules and especially revocation of existing rights. Revocation of rights is simple and guaranteed compared to the earlier mechanisms and is done the same way as the addition of new rules. The policy change propagation is facilitated with a new primitive `put_policy(key, access list)` which is routed to the responsible node which replaces the old ACF rule for the resources, with the new one. However, to be noted, the same primitive is used for the case of revocation of an ACF rule also. However, when a new `put_policy()` request arrives at a node, it must check whether the updating node is the owner of the resource or not. At any point of time, only owner nodes can change the ACF rule associated with a resource. This is possible by caching the identity of the owner node when an index entry is first time published into the network with `put()` request. Later on the identity of the updating node should be checked against this cache so as to allow only owner nodes to update the ACF. The replication mechanism is assumed to maintain consistency across replicas of index or resource entries.

Regarding updates made to TSO w.r.t the membership, when a new node joins a TSO, it shares the usual query, storage loads as dictated by the overlay management mechanism used in the TSO. When a node is excluded from a TSO due to either the member is no longer interested to execute the ACF of the TSO members or the member is found to be breaching the faithful execution of the ACF as found out by out-of-band mechanisms. In any case, the TSO must be reconstructed excluding the member. The excluded member then can pose queries for the data hosted in the TSO like any other non-member peer in the system.

**Analysis:** The CR-TSO approach provides high configurability to the applications in regard to number of nodes in a TSO, so that the TSO sizes can be limited, by which, the degree of vulnerability of the data published is limited. By selectively choosing which peers can host their index, the mechanism provides a high degree of faithful ACF execution. Different P2P protocols can be used for overlay and TSO management, which is interesting to study the arising issues and about the right combination of protocols.

On the surface, the CR-TSO approach seems to be analogous to DPN approach where the former tries to enforce ACF by localizing data items only to sub overlay networks created on top of a big overlay, where as the latter does by independent overlays. The CR-TSO approach can be interpreted as an improvement over DPN approach, both share the same philosophy of modeling access control problem as a variation of admission control problem. In DPN, we control the members of the whole network where as in CR, the members of a community are controlled. However, the CR-TSO approach promises some advantages which are not possible to be realized with DPN approach. Hence, it is highly preferred over DPN for realizing ACPeer systems.

- One major advantage with CR-TSO is a node which is not member of TSO can also issue a query for the data hosted in the TSO. The Trusted reference monitor implemented at each TSO will reply to such queries accordingly. What all a peer needs in order to search for a resource, is the TSO\_ID of the TSO it is interested

in. In the case of DPN, a peer must be part of the network and hence should participate in all the roles of the network.

- Consider a scenario where a peer has access to only one resource shared in the network. In the case of DPN, the peer must be part of the network and thus has access to all the resources shared which leads to huge amount of information leakage. In the case of CR, the peer need not be part of the TSO and still can be allowed to access this single resource alone. It clearly demonstrates that CR based approach promises increased easiness in managing the access control framework.
- This is advantageous even when a node's access permissions are revoked. In the case of DPN, the entire P2P network need to be built again excluding the peer. In the case of CR, only the policy is updated on the peers in the TSO unless the excluded peer is one of the members of the TSO itself.
- CR approach is inherently more scalable than the DPN because in the case of CR-TSO, the applications can flexibly change ACF without altering the network state as such, and as described earlier, it is highly configurable which is a desired property for any networked system.

**Scope for Improvements:** However, as described in the following, the CR-TSO approach lacks on some fronts. Since the published data is localized inside a TSO only, which has normally a fewer number of nodes, such data does not match the availability guarantees possible in the huge public overlay due to high number of peers. In general, increased availability of resources in the light of peer failures is achieved with the technique of replication. Thus, a smaller sized TSO can not match the overlay w.r.t the level of geographical diversity and the number of peers it provides which are desirable characteristics for any high available replication technique.

Further, management of sub overlay may add further overhead to underlying overlay management. Overlay management overhead includes the cost of maintaining consistent routing table entries, cost of addition or deletion of a node to/from the overlay. In the following sub section, we propose a light weight approach to the CR-TSO case, where instead of constructing sub overlays on top of the overlay, the members of a TSO assume a proxy role for the queries posted onto the TSO. TSO's published data is stored on the big overlay with the help of encryption mechanisms.

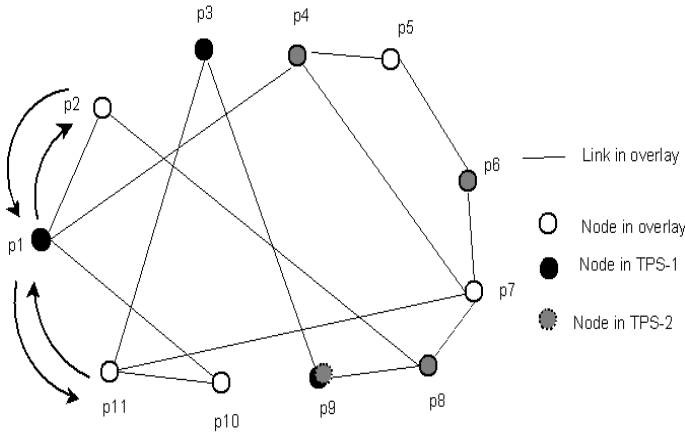
### 4.3.2 CR with Trusted Proxy Set (TPS)

CR-TPS is a light weight approach to the CR based Level-2 system. It nullifies the overhead involved in TSO overlay management by completely getting the rid of the overlay, where the members of a TSO instead of forming an overlay as such, will just assume the role of a set of trusted proxies with each member of the set acting as a proxy for the queries targeted at the TSO. Thus each TPS is equivalent to a TSO w.r.t the set of member nodes. Any potential member of a TSO can also be a potential member for the corresponding TPS. The same semantics of entry points apply to the case of TPS also where a subset of TPS members announce

themselves as the entry points/gateways for the TPS, however, with a slight variation of the role/duty assumed by the member peers that do not announce themselves as entry points. CR-TPS approach inherits from the CR-TSO approach, the same flexible way of ACF enforcement and wider availability of storage, from the CQ approach.

The concept of TPS and mechanism of searching is illustrated in Figure. 3, where each TSO in Figure. 2 is modeled as a TPS.

**Publishing and Searching:** On contrary to CR-TSO approach, member nodes make use of the storage available on the main overlay, instead of the storage within the sub overlay only, for storing ACData items published by them. Before publishing, the data is encrypted with secret keys known only to the members of the TPS. In this case, the simplest key management approach is using a single secret key for every resource published by the TPS members. This secret key is defined when the TPS is initially bootstrapped. So any new member who joins the TPS later simply gets access to this key. A searching peer first contacts one of the entry points to the TPS and forwards the query to the node as done in the case of CR-TSO case. The contacted node converts the query into equivalent encrypted query with the corresponding encrypted resource identifier and inserts the query onto the main overlay. Once the node gets the reply, it decrypts the result and sends to the original querying node. This way, the members of the TPS act just as proxies for the data published by the members. In this simple key management technique, the members which did not announce themselves as entry points, do not have any role in the search phase, because such peers will never be contacted by the peers for searches.



- (1) p11 queries TPS-1
- (2) p1 proxies for the query, and requeries the network- received by p2
- (3) p2 sends reply to p1
- (4) p1 replies to original querying peer p11

**Figure 3: Illustration of Trusted Proxy Set.**

However, this simple key management technique leaves the entire ACData published vulnerable in the case of the key leakage to unauthorized peers. Hence, several keys can be used for encrypting the ACData items with list of mappings of data item and the key stored at every entry point. In this case also, the member nodes which do not contribute to the entry point list will not participate in search phase.

When a peer is excluded from a TPS (for the same rea-

---

**Algorithm 4** Simplified algorithms for CR-TPS

---

- 1: {The algorithms are simplified for brevity. Each algorithm is run at each peer  $p$  in the system.}
  - 2: *Publish()*
  - 3: {The algorithm assumes that  $p$  selects a single TPS for hosting all of its resources.}
  - 4: choose a TPS  $t$  out of all known TSOs such that  $L(t)$  is minimum
  - 5: **for all** resource  $r \in \mathcal{R}^p$  **do**
  - 6:   encrypt  $ACData(r)$  with secret key of TPS  $t$  and publish into main overlay
  - 7:   broadcast  $((ACF^p)^{-1})(r)$ , into  $t$
  - 8:   {assuming that each TPS member maintains ACF of all TPS members}
  - 9:   **if**  $Level - 1.5$  access control is the requirement for  $r$  **then**
  - 10:     publish  $(key, TPS\_Id)$  pair into main overlay using  $put()$  primitive
  - 11:   **end if**
  - 12: **end for**
  - 13: Optionally, join *entry point list* of the TPS
  - 14: *Search()*
  - 15: **if** The TPS to search in, is already decided **then**
  - 16:   {either using *data descriptors* OR interpreting the TPS identifier OR from the list of TPS identifiers retrieved by querying main overlay for interested *key*}
  - 17:   retrieve *entry point list* of the TPS
  - 18:   send query message to one entry point for the interested *key*
  - 19: **end if**
  - 20: **if** the TPS to search in, is not decided **then**
  - 21:   repeat above in each TPS known
  - 22: **end if**
-

sons discussed for the case of TSO), any new data published after that should be encrypted with new keys which are not shared with the excluded member. However, the excluded member will be still able to access data published with old keys. To minimize the amount of such data, instead of every member knowing about all the keys used to publish the data, peers share part of the identifier space and the corresponding keys. This approach introduces the overlay among the TPS members, which in turn will be analogous to the TSO model. In this model, members that are not entry points will also participate in the search phase, as the entry points have to contact them for any search queries related to the identifier space shared by those peers.

Analysis: CR-TPS approach frees the system from the need to manage sub overlays. By building minimal framework on top of the main overlay technology, it tries to realize a Level-2 system. To serve search requests, only one member peer from the entire TPS, is enough to be online, which is not possible with CR-TSO. Since items are not stored on the TPS nodes as such, a member can know all of the items published by TPS members, only by querying the main overlay. On the other hand, in CR-TSO, the items are stored on the nodes and hence exposed to the peers by default. The kind of leakage in the former case can be termed as *hard leakage* where as the latter one as *soft leakage*. Hard leakage is a more desirable property of an ACPeer system than a soft leakage.

Another promising characteristic of TPS approach is, the framework can be augmented in such a way, multiple peers together can decide on access control requests in stead of a single peer as demonstrated above. This can be done using threshold cryptography where a single secret key can be split into multiple parts and shared with multiple peers of a TPS, more than one share is required to construct the original key.

#### 4.4 Hybrid Approach

This approach tries to mix both CQ and CR approaches together- by encrypting the TSO's entry points list and limiting access to only the peers who have the key- thus ensuring a CQ-like ACF enforcement on the metadata used for TSO/TPS discovery and a CR-like ACF enforcement for the ACData hosted inside the corresponding TSO/TPS. This way, the system ensures that not every peer in the main overlay contacts any TSO/TPS, at the same time ensuring that access to ACData item is controlled.

#### 4.5 Comparison Analysis

All the approaches discussed in the paper are compared against each other w.r.t the following criterion in the Table. 1.

- *Publication cost* measures the number of times the same single ACData item must be published based on the number of principals allowed to access. For some solutions, the same item must be published once for each principal where as for others, only one instance is published for all the principals in the system.
- *Search cost* measures the overhead due to access control, in terms of number of search queries an arbitrary peer has to pose to retrieve an interested ACData item. For a Level-0 system, it is always 1.
- The overhead involved in mitigating with ACF updates and revocation are captured in the respective criterion.

- *Additional requirements* capture the potential additional requirements imposed by the approach for deployment apart from the requirement of strong identities which is a common requirement for all approaches. It should be noted that some simple solutions were described in the paper for some of the requirements.
- Some approaches have inherent leakage of some ACData to unauthorized peers as part of the design. This is captured in respective metric.
- *Storage capacity available* measures the number of peers available at disposal for storing a published ACData item, is assumed to capture the level of fault tolerance available in the system, which is captured in the last criterion.

## 5. CONCLUSIONS

## 6. REFERENCES

- [1] R. Steinmetz and K. Wehrle, "Peer-to-peer systems and applications," *Lecture Notes in Computer Science*, 2005.
- [2] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A scalable Peer-To-Peer lookup service for internet applications," in *Proceedings of the 2001 ACM SIGCOMM Conference*, 2001, pp. 149–160. [Online]. Available: [citeseer.ist.psu.edu/stoica01chord.html](http://citeseer.ist.psu.edu/stoica01chord.html)
- [3] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," *Lecture Notes in Computer Science*, vol. 2218, pp. 329–340, 2001. [Online]. Available: [citeseer.ist.psu.edu/rowstron01pastry.html](http://citeseer.ist.psu.edu/rowstron01pastry.html)
- [4] K. Aberer, "P-Grid: A self-organizing access structure for P2P information systems," *Sixth International Conference on Cooperative Information Systems (CoopIS 2001), Lecture Notes in Computer Science*, vol. 2172, pp. 179–194, 2001. [Online]. Available: [citeseer.ist.psu.edu/aberer01pgrid.html](http://citeseer.ist.psu.edu/aberer01pgrid.html)
- [5] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content addressable network," in *ACM SIGCOMM 2001*, August 2001.
- [6] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's highly available key-value store," in *21st ACM Symposium on Operating Systems Principles, SOSP'07*, October 2007.
- [7] Yahoo! Research, "Pnuts- platform for nimble universal table storage," <http://research.yahoo.com/node/212>.
- [8] "Wuala- the p2p storage," <http://wua.la/en/home.html>.
- [9] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-area cooperative storage with cfs," in *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*. New York, NY, USA: ACM, 2001, pp. 202–215.
- [10] M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows, and M. Isard, "A cooperative internet backup scheme," in

- ATEC '03: Proceedings of the annual conference on USENIX Annual Technical Conference.* Berkeley, CA, USA: USENIX Association, 2003, pp. 3–3.
- [11] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer, “Farsite: federated, available, and reliable storage for an incompletely trusted environment,” in *OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation.* New York, NY, USA: ACM, 2002, pp. 1–14.
- [12] “Nepomuk: The social semantic desktop,” <http://nepomuk.semanticdesktop.org/>.
- [13] “Team: Tightening knowledge sharing in distributed software communities by applying semantic technologies,” <http://www.team-project.eu/>.
- [14] K. Aberer, P. Cudre-Mauroux, M. Hauswirth, and T. van Pelt, “GridVine: Building internet-scale semantic overlay networks,” in *International Semantic Web Conference (ISWC)*, ser. LNCS, vol. 3298, 2004, pp. 107–121. [Online]. Available: [citeseer.ist.psu.edu/aberer04gridvine.html](http://citeseer.ist.psu.edu/aberer04gridvine.html)
- [15] M. Karnstedt, K.-U. Sattler, M. Richtarsky, J. Müller, M. Hauswirth, R. Schmidt, and R. John, “Unistore: Querying a dht-based universal storage.” in *ICDE*. IEEE, 2007, pp. 1503–1504.
- [16] M. Cai and M. Frank, “Rdfpeers: a scalable distributed rdf repository based on a structured peer-to-peer network,” in *WWW '04: Proceedings of the 13th international conference on World Wide Web.* New York, NY, USA: ACM, 2004, pp. 650–657.
- [17] Grumbacher and Nuremberg, “Posix access control lists on linux,” <http://www.suse.de/agruen/acl/linux-acls/online>.
- [18] “The p-grid project,” <http://www.p-grid.org/index.html>.
- [19] N. Rammohan, “A note on secure p-grid,” *Technical Report, LSIR-REPORT-2008-005, Swiss Federal Institute of Technology (EPFL)*.
- [20] R. S. Sandhu and P. Samarati, “Access control: Principles and practice,” *IEEE Communications Magazine*, vol. 32, no. 9, pp. 40–48, 1994. [Online]. Available: [citeseer.ist.psu.edu/sandhu94access.html](http://citeseer.ist.psu.edu/sandhu94access.html)
- [21] A. Singh, “Secure management of networked storage services: Models and techniques,” *Ph.D Thesis, College of Computing, Georgia Institute of Technology, 2007*.
- [22] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, “Plutus: Scalable secure file sharing on untrusted storage,” in *FAST '03: Proceedings of the 2nd USENIX Conference on File and Storage Technologies.* Berkeley, CA, USA: USENIX Association, 2003, pp. 29–42.
- [23] E. Goh, H. Shacham, N. Modadugu, and D. Boneh, “SiRiUS: Securing Remote Untrusted Storage,” in *10th Annual Network and Distributed System Security Symposium (NDSS)*, 2003.
- [24] J. Li, M. Krohn, D. Mazières, and D. Shasha, “Secure untrusted data repository (sundr),” in *OSDI'04: Proceedings of the 6th conference on Symposium on Operating Systems Design & Implementation.* Berkeley, CA, USA: USENIX Association, 2004, pp. 9–9.
- [25] J. F. da Silva, L. P. Gaspary, M. P. Barcellos, and A. Detsch, “Policy-based access control in peer-to-peer grid systems,” in *GRID '05: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing.* Washington, DC, USA: IEEE Computer Society, 2005, pp. 107–113.
- [26] P. Fenkam, S. Dustdar, E. Kirda, G. Reif, and H. Gall, “Towards an access control system for mobile peer-to-peer collaborative environments,” in *WETICE '02: Proceedings of the 11th IEEE International Workshops on Enabling Technologies.* Washington, DC, USA: IEEE Computer Society, 2002, pp. 95–102.
- [27] K. Watanabe, Y. Nakajima, N. Hayashibara, T. Enokido, M. Takizawa, and S. M. Deen, “Trustworthiness of peers based on access control in peer-to-peer overlay networks,” in *ICDCSW '06: Proceedings of the 26th IEEE International Conference Workshops on Distributed Computing Systems.* Washington, DC, USA: IEEE Computer Society, 2006, p. 74.
- [28] Y. Zhang, X. Li, J. Huai, and Y. Liu, “Access control in peer-to-peer collaborative systems,” in *ICDCSW '05: Proceedings of the First International Workshop on Mobility in Peer-to-Peer Systems (MPPS) (ICDCSW'05).* Washington, DC, USA: IEEE Computer Society, 2005, pp. 835–840.
- [29] B. Crispo, S. Sivasubramanian, P. Mazzoleni, and E. Bertino, “P-hera: Scalable fine-grained access control for p2p infrastructures,” in *ICPADS '05: Proceedings of the 11th International Conference on Parallel and Distributed Systems (ICPADS'05).* Washington, DC, USA: IEEE Computer Society, 2005, pp. 585–591.
- [30] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, “Freenet: A distributed anonymous information storage and retrieval system,” *Lecture Notes in Computer Science*, vol. 2009, pp. 46–67, 2001. [Online]. Available: [citeseer.ist.psu.edu/clarke00freenet.html](http://citeseer.ist.psu.edu/clarke00freenet.html)
- [31] A. Singh, B. Gedik, and L. Ling, “Agyaat: Mutual anonymity over structured p2p networks,” *Emerald Internet Research Journal*, vol. 16, no. 2, 2006.
- [32] V. Scarlata, B.N.Levine, and C. Shields, “Responder anonymity and anonymous peer-to-peer file sharing,” in *ICNP '01: Proceedings of the Ninth International Conference on Network Protocols.* Washington, DC, USA: IEEE Computer Society, 2001, pp. 272–280.
- [33] R. Dingledine, M. J. Freedman, and D. Molnar, “The free haven project: distributed anonymous storage service,” in *International workshop on Designing privacy enhancing technologies.* New York, NY, USA: Springer-Verlag New York, Inc., 2001, pp. 67–95.
- [34] M. K. Reiter and A. D. Rubin, “Crowds: anonymity for web transactions,” *ACM Trans. Inf. Syst. Secur.*, vol. 1, no. 1, pp. 66–92, 1998.
- [35] P. F. Syverson, D. M. Goldschlag, and M. G. Reed, “Anonymous connections and onion routing,” in *SP '97: Proceedings of the 1997 IEEE Symposium on Security and Privacy.* Washington, DC, USA: IEEE Computer Society, 1997, pp. 44–54.
- [36] “Gnutella,” <http://www.gnutella.com>.
- [37] “Kazaa,” <http://www.kazaa.com>, 2002.

- [38] "Ssl specification," <http://wp.netscape.com/eng/ssl3/>.
- [39] T. Dierks and E. Rescorla, "Rfc 4346-the transport layer security (tls) protocol, version 1.2," <http://www.ietf.org/rfc/rfc4346.txt>.
- [40] R. Miller, "More p2p traffic using ssl encryption," <http://www.datacenterknowledge.com>.
- [41] S. Krasinsky, "Project jxta overview," <http://www.mnlab.cs.depaul.edu/seminar/spr2004/JXTAOverview.pdf>.
- [42] N. Krishnan, "Jxta and security," *Technical Report*, Sun Micro Systems, <http://java.sun.com/developer/Books/networking/jxta/jxtap2pch08.pdf>.
- [43] B. Yeager, "P2p security and jxta," *Technical Report*, Sun Labs, <http://www.ida.liu.se/conferences/p2p/p2p2003/presentations/Yeager.pdf>.
- [44] M. Bawa, J. Roberto J. Bayardo, and R. Agrawal, "Privacy-preserving indexing of documents on the network," in *vldb'2003: Proceedings of the 29th international conference on Very large data bases*. VLDB Endowment, 2003, pp. 922–933.
- [45] T. Luu, F. Klemm, I. Podnar, M. Rajman, and K. Aberer, "Alvis peers: a scalable full-text peer-to-peer retrieval engine," in *P2PIR '06: Proceedings of the international workshop on Information retrieval in peer-to-peer networks*. New York, NY, USA: ACM, 2006, pp. 41–48.
- [46] A. Singh, M. Srivatsa, and L. Liu, "Efficient and secure search of enterprise file systems," in *IEEE International Conference on Web Services (ICWS 2007)*. IEEE Computer Society, July 2007, pp. 18–25.
- [47] Y. Kim, D. Mazzocchi, and G. Tsudik, "Admission control in peer groups," in *NCA '03: Proceedings of the Second IEEE International Symposium on Network Computing and Applications*. Washington, DC, USA: IEEE Computer Society, 2003, p. 131.
- [48] N. Saxena, G. Tsudik, and J. H. Yi, "Admission control in peer-to-peer: design and performance evaluation," in *SASN '03: Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*. New York, NY, USA: ACM, 2003, pp. 104–113.
- [49] B. Schneier, "Applied cryptography, second edition: Protocols, algorithms, and source code in c," 1996.
- [50] N. J. A. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman, "Skipnet: a scalable overlay network with practical locality properties," in *USITS'03: Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems*. Berkeley, CA, USA: USENIX Association, 2003, pp. 9–9.
- [51] W. Galuba, K. Aberer, Z. Despotovic, and W. Kellerer, "The complex facets of reputation and trust," in *FuzzyDays06*, 2006.

**Table 1: Comparison chart**

Criterion	Approach					
	Any Level-0 system	DPN	CQ	CR-TSO	CR-TPS	Hybrid
<i>Publication cost</i>	Only once	Once per each authorized principal OR only once depending on the optimization criterion	Once per each authorized principal	Only once	Only once	Only once for the AC-Data items and once per each principal for the TSO entry point list
<i>Search cost</i>	Only one search query for an interested item	Only once OR One search query per member principal depending on the optimization criterion	For Level-1.5, as many queries as the number of accessible principals. For Level-2 system, it is number of principals * number of potential publishers	For Level-1.5, 1+ no. of TSOs selected for search. For Level-2, twice the no. of TSOs selected to search in (once for the entry point list and once for the item search)	3*(no. of selected CR-TSOs) since one more query for each search request because of extra indirection	#cost of CQ + no. of TSOs selected to search in
<i>Policy updates and revocation</i>	N/A	Revocation needs reconstructing the entire private network. Addition of a new rule may require to create a new private network.	Revocation requires removal of old encrypted data and republishing with new keys. Addition of a new rule is just sharing the key with new set of principals.	No publication cost in either case. Just the policy must be updated with in the TSO.	Same as CR-TSO.	#cost of CQ + #cost of CR
<i>Additional Requirements</i>	N/A	Admission control protocols.	Key management and distribution mechanism.	A light weight admission control with lesser no. of peers.	A light weight admission control with lesser no. of peers. Secret key management strategy.	Requirements for CQ and CR approaches.
<i>Leakage to unauthorized peers inherent to the system</i>	N/A	Zero	Zero	Zero with appropriate members in a TSO. Non-zero but with in tolerable limits, in normal case.	Same as CR-TSO	Same as CQ and CR together.
<i>Available storage capacity</i>	Entire network of peers.	Limited by the size of the private network which is orders of magnitude smaller than main overlay	Entire network of peers	Limited by the size of TSO which is orders of magnitude smaller relative to main overlay	Entire network of peers	Depends on whether CR-TSO or CR-TPS is used.