

Towards Access Control Aware P2P Data Management Systems

Rammohan Narendula
EPFL, Switzerland
rammohan.narendula@epfl.ch

Zoltán Miklós
EPFL, Switzerland
zoltan.miklos@epfl.ch

Karl Aberer
EPFL, Switzerland
karl.aberer@epfl.ch

ABSTRACT

P2P data management systems provide a scalable alternative to centralized architectures. Their adoption, however, is limited by the lack of possibility to control the access to the resources stored in the system. We address this problem in the case of structured P2P networks, in particular, when the system is used in a collaborative working environment. We analyze the problem assuming a simple threat model and we systematically explore the solution possibilities. We design and compare access control enforcement techniques which realize the desired functionality by constructing independent networks or by implementing access control at query or at response time.

1. INTRODUCTION

The "Peer-to-Peer" communication model has introduced a significant paradigm shift in the way users share the resources and communicate with each other on the Internet. Structured P2P systems based on Distributed Hash Table (DHT) paradigm are increasingly adopted in building massively scalable and highly available data management systems. The academic research community has pursued a wide range of such systems including Chord, Pastry, and CAN [12]. DHT based scalable storage infrastructures are embraced beyond the realms of academia also [1].

In spite of increased usage, the lack of handling privacy and security issues sets clear limits to the adoption of these systems. The most important concerns include *securing the communication infrastructure* and *securing the access to the shared resources*. The first concern requires that no third parties can intercept or change the messages between any two peers. The latter ensures that only authorized peers have access to shared resources.

In this paper, we study ways to enhance existing P2P systems to provide means to enforce access control. We refer to an access control aware P2P data management system as *ACPeer* throughout the paper. The paper studies *ACPeer* systems based on structured P2P networks where, for each

resource r , an index item which is in the form of $(key, value)$ pair, is stored in the system. We argue that the index information is also a valuable resource to be protected. For instance, in the case of DHT based RDF triple stores, like the GridVine [2] system, the RDF triples form the value field of index, which may contain sensitive metadata. Therefore, our goal is to realize a system, where one can grant read access to both the resources and the index items. For example in TEAM project, the GridVine network is used to share the knowledge of software engineers, represented as RDF triples. The peers publish RDF triples to the GridVine network. An RDF triple of the form (s, p, o) is indexed 3-times to enable lookup queries containing the subject, the predicate or the object of a triple. Thus, the RDF triples themselves reside in the index, in the value fields of the index.

In the case of a P2P system the enforcement of the access rights can be realized at various points. We designed our methods by carefully choosing the enforcement points and the methodology, to keep administration and processing costs low. We begin with a trivial approach which realizes *ACPeer* systems by constructing independent networks. Then we introduce an encryption-based technique which embeds access control into the index generation itself. Later, we propose a technique of enforcing access control at query reply time.

The remaining of the paper is organized as follows. Section 2 gives a more precise definition of the access control and motivating scenarios. In Section 3, we describe and compare the various techniques for realizing access control in structured P2P networks. Section 4 discusses related work and finally Section 5 concludes the paper with some insights into future work.

2. OVERVIEW OF THE PROBLEM

2.1 Access control in structured P2P networks

In a structured P2P network the resources are found using indexing where the index items, in the form of $(key, value)$ pairs, are stored in and retrieved from the network using two system provided primitives namely `put(key, value)` and `get(key)` respectively. The attribute `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).

Figure. 1 illustrates the typical work flow of a structured P2P system. A peer F wishes to publish a resource with

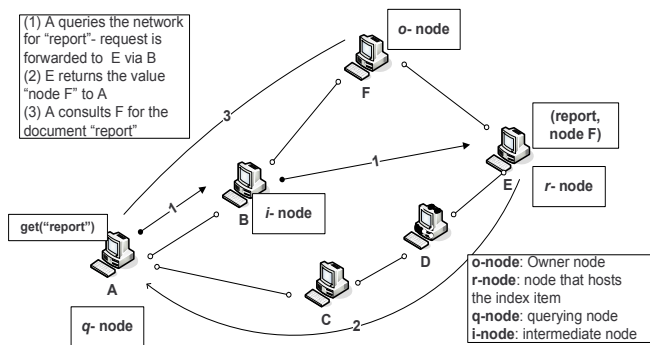


Figure 1: A typical message flow in a structured P2P system.

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*).

Often, in any P2P infrastructure, more than one user at application level, can access a single peer at the overlay level. In this paper we assume existence of a single user per peer and hence, we use the terms *user*, *peer* and *node* interchangeably.

In an ACPeer system, users publish resources and associate access control rules with the resources, which define who in the network is authorized to access the resource.

We distinguish various levels of access control in the following way. If a system does not provide access control at all, we call it a *Level-0 system*. If it intends to protect the resources, but not their indexes, we call it a *Level-1 system*. For instance, as an analogy, the IEEEExplore¹ system where the metadata of each document is open but the document is available to only paid authorized users, would be a Level-1 system. If it intends to protect for each resource both the resource and its index, we call it a *Level-2 system*. If the system protects the resources, but only the key-field of their index (but not the value), then we call it *Level-1.5 system*. In a Level-1.5 system, an arbitrary peer can find out whether a resource with certain id exists in the system or not, without being able to obtain any further information.

Our goal is to enhance existing structured P2P networks to realize a Level-2 ACPeer system.

2.2 Modeling Access Control Policy

In this work, we pursue discretionary access control model (DAC). We assume that each resource has an owner who stores the resource and defines the access control rules. We do not assume any particular structure of the resources and use a flat model without any hierarchies. One could consider extending our flat resource model in order to support hierarchies. We call the subjects to whom access rights are granted as *principals*. We assume a general modeling of a principal, but the solutions can be adopted to different types

of principals, such as principals modeled as users and groups of users, principals based on roles in an organization, principals modeled using relations on social graph.

The access rights are expressed in the form of access control lists (ACL). We use the notation \aleph for the set of principals, \mathcal{R}^p for the set of resources owned by the peer *p*, $ACData(r)$ for the set of protected components for a resource *r*. For example, in a Level-2 system, for each resource *r*, $ACData(r) = \{r, index(r)\}$.

$$ACL^p : \bigcup_{r \in \mathcal{R}^p} ACData(r) \rightarrow 2^{\aleph}$$

For simplicity, we assume that a resource *r* and the $index(r)$ will have the same access control rights. The solutions do support the other case where each will have its own rights.

Thus, the goal of a Level-2 ACPeer system to enforce the ACL of all peers in the system. An agent that executes the ACL on a node is referred as *reference monitor*.

2.3 Assumptions

We assume that the resources associated with an index item, if any, are stored with the corresponding *o-nodes*, unless specified otherwise. When an ACData item is said to be published, we are referring to publishing index. Thus the *value* part of the index information of the form $(key, value)$ points to the location of the *o-node*. Our solutions can be adapted to other storage models, which is not discussed here.

Since establishing identity of a principal securely is a critical prerequisite for any access control system. We assume the existence of a secure mapping function which maps an access request to one of the principals in the system, using which the reference monitor can either allow or deny the access request based on the policy statement. For some of the solution approaches, we assume the existence of a public-key infrastructure PKI².

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. We argue that this simple threat enables us to concentrate on only the access control aspects of data management and addressing every possible compromise of the system is beyond the scope of the paper.

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 the 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 must be protected for privacy reasons. 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.

3. SOLUTION MECHANISMS

In this section, we explore the problem of designing a Level-2 ACPeer system in detail, introduce the following mechanisms to solve the problem.

1. Independent P2P Networks (IPN)
2. Controlled Queries (CQ)

¹<http://www.ieeexplore.org>

²<http://www.ietf.org/html.charters/pkix-charter.html>

Algorithm 1 Simplified algorithms for IPN

{The algorithms for a peer p in the system:}
PROCEDURE: PUBLISH()
for all resource $r \in \mathcal{R}^p$ **do**
 publish $ACData(r)$ into a single network of all principals in, *or* networks each with one principal from, $ACL^p(r)$
end for
PROCEDURE: SEARCH()
for all every network with valid membership **do**
 search for the item in the network
end for

3. Controlled Replies (CR)

The IPN approach addresses the problem by constructing independent networks. The CQ approach realizes the reference monitors in the network using encryption techniques whereas CR, in an encryption- free manner.

3.1 Independent P2P Networks (IPN)

A simple and trivial way of realizing a Level-2 system is by creating a set of independent P2P networks each with its own resources published and the peers participating. We argue that in some specific cases this naive solution could be useful. There are two ways to create a new network which depend on which one of the search or publication costs has to be optimized. The IPN approach is explained in detail in [9].

However, IPN approach has inherent scalability and efficiency problems, as the number of networks needed would explode exponentially. As it is discussed in [9], the number of networks can be in the order $O(|\Theta| 2^{|\mathcal{N}|})$, where Θ denotes the set of peers in the system, or $O(|\Theta| |\mathcal{N}|)$, depending on our choice of resource allocation strategies. The independent networks can become highly unmanageable if the number of resources and peers increase.

The IPN solution tries to attack the problem of Level-2 ACPeer system by realizing the reference monitors around *o-nodes* (refer Figure. 1) only and hence end up in non-scalable systems. By designing a *q-node* or *r-node* centered reference monitors, one can realize scalable and manageable solutions as demonstrated in next sections where the techniques of *controlled queries* and *controlled replies* are introduced.

3.2 Controlled Queries (CQ)

The intuition behind the CQ based approach is to embed reference monitors into the index generation process itself by using encryption techniques and ensuring that only the authorized principals get access to the encryption keys. Such encrypted resource or index 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. Resource identifier encryption schemes were used for censorship-resistant publishing in P2P systems [4].

Based on the encryption keys used, either a Level-1.5 or Level-2 CQ based ACPeer system can be realized. In the following, we first introduce a solution to realize a Level-1.5 system and then deal with a Level-2 system using additional encryption keys and increased search cost. Level-1.5 ACPeer System- Publishing and Searching: A Level-1.5 system is realized using public key cryptography: each

Algorithm 2 Simplified algorithms for CQ

{The algorithms for a peer p in the system:}
PROCEDURE: PUBLISH_Level1.5()
for all resource $r \in \mathcal{R}^p$ **do**
 for all principal $n \in ACL^p(r)$ **do**
 encrypt *key* of $index(r)$ with a deterministic and *value* with a non-deterministic encryption algorithm using public key of n
 publish encrypted $index(r)$ pair into the network
 end for
PROCEDURE: SEARCH_Level1.5()
for all member principal **do**
 query for *key* encrypted with public key of the principal using the deterministic encryption algorithm
 decrypt the result with private key
end for
PROCEDURE: PUBLISH_Level2()
for all resource $r \in \mathcal{R}^p$ **do**
 encrypt $index(r)$ with a single secret key for all principals in, *or* with a separate secret key for each principal in $ACL^p(r)$
 publish into the network
end for
PROCEDURE: SEARCH_Level2()
for all known/acquired secret key **do**
 query for interested item key encrypted with secret key
 decrypt the result
end for

principal in the system possesses a public, private key pair. For an authorized principal, the ACData item is inserted into the network after encrypting it with the principal's public key. So a peer can search in all the items published to itself by encrypting the item's identifiers with public key. In fact, the system allows any peer knowing the principal's public key to search, but the results can be interpreted only by the peers which have the respective private key.

In order to preserve the searchability of the index we propose to encrypt the *key* part of index with a deterministic encryption algorithm³ [10]. Hence, all peers can produce the same encrypted identifier for a resource originally used by the publisher. However, using deterministic encryption for both the fields of an index entry, will lead to dictionary attacks on the value field, thus degrading the system to Level-1, as explained in detail in [9]. Hence, to achieve a Level-1.5 system requirements (i.e., preventing leakage of *value* field of index), we propose to encrypt the *value* field by a non-deterministic algorithm.

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 item. This overhead is in the order of $O(|\mathcal{N}|)$. 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

³A deterministic encryption ensures that a given plain text and key combination always compute to the same cipher text every time the algorithm is applied, whereas, in a non-deterministic case, a different cipher text results every time the algorithm is applied. However, all such ciphers decrypt to the same plain text.

for each principal, for which, the peer has credentials (public and private key pairs) for. Hence, number of search queries is also in the order of $O(|\mathcal{N}|)$.

Level-2 ACPeer System- Publishing and Searching: A CQ-based Level-2 system is realized by encrypting the ACData items with a secret key (instead of public key) known only to the publisher and the authorized principals. The secret key generation process which can be done in two ways, dictates the publication overhead and the number of keys required. The publisher generates a single secret key for all the authorized principals for a resource and publish ACData with this secret key. Alternatively, a key per each principal is generated and the ACData item is published separately for each authorized principal. The secret keys are distributed by offline mode or by encrypting with the principal's public key and publishing into the network, which is discussed more in [9]. The authorized peer queries the system for a search key 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 $O(|\Theta| 2^{|\mathcal{N}|})$ or $O(|\Theta| |\mathcal{N}|)$ depending on the two cases discussed above⁴. In addition, the search overhead also increases significantly compared to Level-1.5 system, as the number of search queries required to search for an interested item would be in the order number of secret keys.

Revocation: Revocation of access rights 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 [9]. ACData items published with old credentials are removed from the network and republished with new credentials, when a revocation is requested. Given the fact that the ACData items are stored on arbitrary untrusted peers, implementing a reliable deletion operation is challenging.

CQ based approach is promising as it does not introduce any changes to the routing protocol of the underlying P2P system. An access-controlled query works like any other query and processing the requests is quite transparent to the protocols, and hence can be applied on top of any existing structured P2P system. CQ approach seems to be more suitable for a Level-1.5 than a Level-2 system and by sharing the public keys in a restricted way, one can assume that CQ-Level-1.5 provides more controlled access on the *key* field than a Level-1 system. Thus, the proposed Level-1.5 system can be assumed to meet a Level-2 system's requirements to a greater extent.

In the next section, we demonstrate the Controlled Replies (CR) approach based Level-2 system which pushes the ACL 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.

3.3 Controlled Replies (CR)

In this approach, ACData items are stored in an encryption-free manner, hence, any peer including unauthorized peers, can post queries for interested items in contrary to the CQ-based approach where peers which can access the encryption keys can only pose queries. CR- based Level-2 system ensures that queries that originate only from the authorized peers are replied i.e. the *r-nodes* implement the necessary reference monitor. The related ACL function is stored

⁴This is analogous to IPN approach.

at *r-node* along with the items. Since access control is exercised at *r-nodes*, we need a trustful mechanism to ensure that a particular *r-node* does so faithfully. In a structured P2P system each 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 faithful reference monitors. To this end, we envision a *constrained indexing* technique, where *o-nodes* control where the ACData items must be stored. In the following sub section, we introduce techniques based on this philosophy.

3.3.1 CR with Trusted Sub Overlays (CR-TSO)

The intuition is that *o-nodes* can choose such a set of peers based on the social relationships and constrain the items to be hosted on these nodes. The underlying paradigm is *"you host and implement reference monitors for my items and I do the same for your items"*. This way, trustworthy communities of peers are formed and members of the community decide together to host on each other, the ACData items and realize the necessary reference monitors. It should be noted that these are the groups of publishers i.e., *o-nodes* and should not be confused with a principal modeled as a group of *q-nodes*.

In this approach, 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, the second as a member of one or more TSOs and hence is responsible for its identifier space as part of the main overlay and that of the affiliated TSOs. A peer publishes any access controlled items into one of the TSOs it is member in, and any open items into the main overlay. A single peer can be member of multiple TSOs. Figure. 2 demonstrates the concept of TSO where two TSOs formed by 3 and 4 peers (TSO-1 and TSO2, respectively), are illustrated. Peer p_9 is member of both the TSOs.

Any resource that is published inside a TSO must be identified globally by both the TSO id and the identifier local to the TSO. We propose to augment the traditional flat identifier space used in conventional P2P systems, with a two dimensional identifier space to identify resources hosted inside TSOs- one dimension for identifying the TSOs and the other dimension for the resources within a TSO. Hence, such resources are identified with a (TSO_ID, resource_ID) pair. Resources that can be publicly accessed by every peer are published into the main overlay, which are identified with flat identifiers. One form of constrained indexing where the resource indexes are localized based on the domain names is presented in [7].

In the following, we assume that TSOs are formed already in the overlay. A more detailed account of TSO formation is given in [9]. Any peer that possesses the properties desirable for trusted reference monitor can be a potential candidate to be included in a TSO. All the ACData items published into a TSO are exposed to all the members of the TSO which we refer as *leakage* and formally quantified later.

An interesting question that arises in CR-TSO is, the routing and management in the sub overlays- whether the same overlay technology used for the main overlay should be used for the sub overlays or a different overlay technology tai-

lored to a TSO needs should be used. A TSO topology can be modeled as a broadcast topology (like in Gnutella) given the small number of nodes in a TSO, whereas 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.

The *Member list* or *Friends list* of a TSO is cached locally on every member peer. Given the size of the TSO, updating caches on all members, when a member joins or leaves a TSO, can be done with a limited communication cost.

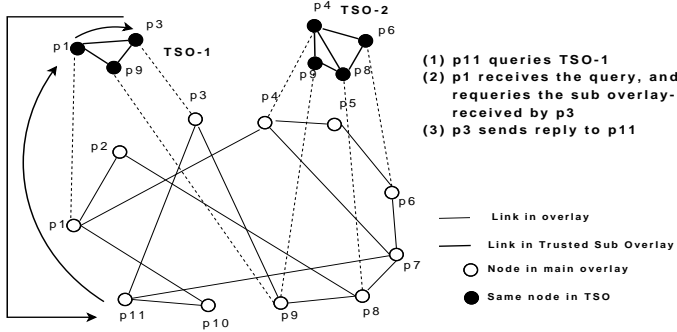


Figure 2: Illustration of Trusted SubOverlays.

Publishing: An ACData item is published into a TSO using TSO-specific primitives for publishing. A publish request message must be routed through only the members of TSO so as prevent any leakage during publishing phase. Once published, an ACData item is visible to potentially every member of the TSO. It should be noted that such member nodes may not be authorized to access the resources as per the ACL. However, we believe that this *leakage*, which is measured in terms of number of peers to whom the items are leaked, 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 assuming all resources in \mathcal{R}^m are hosted inside a single TSO, can be quantified as

$$L^m(t) = \left[\sum_{\forall r \in \mathcal{R}^m} |MemList(t) - ACL^m(r)| \right] - 1$$

where $MemList(t)$ is the Member list of the TSO⁵. Thus, the total amount of *leakage* in a TSO $L(t)$ is leakages of all member peers summed.

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

It should be noted that $L(t) = 0$ if and only if either $|MemList(t)| = 1$ or every member of $MemList(t)$ can access⁵ '1' is because $MemList(t)$ includes the member m itself which is not present in ACF.

cess 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. Alternatively, include 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 smaller number of nodes because TSOs are modeled on social relationships which are limited in number, the amount of leakage is always in tolerable limits.

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. A TSO with minimum leakage is chosen for publishing. Once a TSO is selected, either all or some of the resources are published into the TSO. 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 and contact the TSO members. This is done by publishing a list of *entry points* of a TSO into the main overlay. An entry point is any member node of the TSO which act as a contact point for the TSO. The entry points of a TSO are used in the search process as explained later. Entry point list is a collection of $(TSO_ID, node_ID)$ pairs. The number of entry points in a TSO need not necessarily be equal to number of member nodes of the TSO as some nodes may want to hide their TSO membership information. The authenticity of such an entry point list is of paramount importance for the system to function. One way of realizing it is by using certificate authorities of PKI infrastructure to issue entry point list certificates to all legitimate members of a TSO.

Searching: The best feature of CR-TSO is to allow a peer which is not a member of a TSO to query for the resources inside a TSO and access them if authorized. In CR-TSO, two types of searches exist. An arbitrary peer can search for the resources published into the main overlay using the $get()$ primitive as 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 overlay'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.

However, the important issue arising is, how the q -nodes know about in which TSO to search for when many TSOs exist in the system? 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 likely to be limited when the number of TSOs is bounded. However, if the querying peer knows about the TSO it has to search for, this overhead is limited to two queries- first to retrieve the entry points and then the actual query to this entry point. 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 ACData, can have all the keywords describing the project in the data descriptor. The issue of improved search is not discussed in the paper.

Algorithm 3 Simplified algorithms for CR-TSO

- 1: {The algorithms for a peer p in the system.}
 - 2: *PROCEDURE: 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 $ACL^p(r)$, into t using t 's primitives for publishing
 - 7: [optionally, publish (key, TSO_ID) pair into main overlay using $put()$ primitive]
 - 8: **end for**
 - 9: [Optionally, join *entry point list* of the TSO]
 - 10: *PROCEDURE: SEARCH()*
 - 11: **for all** TSO selected to search in **do**
 - 12: retrieve *entry point list* of the TSO
 - 13: send query message to one entry point for the interested *key*
 - 14: **end for**
-

However, for the index which does not require Level-2 access control, (key, TSO_ID) pairs can be published into the main overlay 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 is analogous to *k-anonymous* solutions where one can attribute a key to a set of k peers (if the size of the TSO is k) but not uniquely to a single peer in this set. At the same time, one can realize a Level-1.5 ACPeer system in this way.

Mitigating updates in ACF and TSO membership: One very promising advantage of CR-TSO approach is the ease to handle the updates in ACL. Both addition of new rules and revocation of rights are simple and dealt the same way using a new primitive `put_policy(key, ACL(key))` which is routed to the responsible node which replaces the old ACL rule for the resources, with the new one. The peers cache the identity of the owners with the resources and check whether the policy updates are originated by the owner or not.

Regarding updates made to TSO 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 ACL of the TSO members or the member is found to be breaching the faithful execution of the ACL 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.

When looked closely, the CR-TSO approach can be interpreted as analogous to IPN approach, but it promises several improvements which are not feasible in IPN approach. A peer which is not member of a TSO can also issue a query

for the items hosted in the TSO. In the case of IPN, a peer must be part of the network and hence should participate in all the roles of the network. In the case of revocation, the entire P2P network need to be built in the case of IPN.

3.4 Comparison Analysis

All the approaches discussed in the paper are compared with each other in Table 1 w.r.t the following criterions. *Publication cost* measures the number of times a single ACData item must be published to realize corresponding ACL rule. For some solutions, each item must be published only once for all the authorized principals, whereas, some solutions it is published as many times as the number of authorized principals. *Search cost* measures the overhead in search, in terms of number of search queries needed to retrieve an interested ACData item. For a Level-0 system, it is always 1. *ACL updates overhead* highlights the overhead involved in mitigating with ACL updates and revocation. *Additional requirements* capture the potential additional requirements imposed by the approach for deployment apart from the requirement of strong identities which is common for all approaches. However, it should be noted that some simple solutions were described in the paper for 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, and is assumed to capture the level of fault tolerance available in the system.

4. RELATED WORK

Any ACPeer system can be assumed to be analogous to a widely-studied problem of hosting files or data on untrusted storage. In [11], the author addresses 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, the data access patterns in structured P2P systems vary from that of these systems, where peers publish index for each resource shared. The searching 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. An access control system for collaborative environments involving mobile and P2P systems is addressed in [6]. An access control framework for P2P overlay networks is defined in [13], where the trustworthiness of acquaintances –on which the access control decision is based– is defined as a modal-logic style satisfiability problem. Modeling access control in the case of P2P collaborative systems is addressed in [14]. 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. Similarly to our work, PHera [5] proposes a 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

Table 1: Comparison chart

Criterion	Approach		
	IPN	CQ	CR-TSO
<i>Publication cost</i>	1 or #authorized principals (depending on the chosen allocation strategy)	#authorized principals	1
<i>Search cost</i>	$O(\Theta 2^{ \mathcal{N} })$ or $O(\Theta \mathcal{N})$ (depending on the chosen allocation strategy)	$O(\mathcal{N})$ (for Level-1.5) or $O(\Theta \mathcal{N})$ (for Level-2)	For Level-1.5: 1+ #TSOs selected. For Level-2: 2*#TSOs selected (because, once for the entry point list and once for the item search)
<i>Policy update overhead</i>	Revocation needs reconstructing the entire private network. Addition of a new rule may require to create a new private network.	Revocation requires a secured <i>remove(key)</i> primitive. Addition of a new rule involves either publishing to new principal or sharing a key with it.	Just the policy must be updated with in the TSO. No extra publishing.
<i>Additional Requirements</i>	Admission control protocols.	Key management and distribution mechanism.	A light weight admission control with a smaller number of peers.
<i>Leakage to unauthorized peers</i>	Zero	Zero	Zero with appropriate members in a TSO. Non-zero, in normal case.
<i>Available storage capacity</i>	Orders of magnitude less than in the main overlay	Entire overlay network	Orders of magnitude less than in the main overlay

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.

ACPeer systems may be assumed to be analogous to anonymous P2P systems. A detailed contrast of both the systems is provided in [9].

Closely related to our research is the problem of privacy preserving indexing, see e.g. [3]. There argue that any indexed document can be constructed with ease from the open inverted indexes of the document. These problems are also present in the P2P information retrieval context, see e.g. [8].

5. CONCLUSIONS AND FUTURE WORK

In this paper, we dealt with the problem of access control in structured P2P networks and provided a set of possible enforcement mechanisms assuming a simple threat model. In our discussions we attempt to systematically explore the solution space. One could design other enforcement methods using a combination of our methods. We also provide a basic comparison of the techniques. Our goal was to enhance existing structured P2P systems with access control rather than designing systems from the scratch.

A systematic qualitative and quantitative evaluation of our methods is an ongoing work. We implemented the controlled queries approach in the context of the TEAM project⁶, where the ACPeer system was built on top of GridVine [2]

⁶<http://www.team-project.eu>

P2P network. Our industrial partners are currently testing the feasibility of this solution.

We are planing to develop more advanced search mechanisms for the CR-TSO case as part of our future study. Another challenge is to design an approach which inherits the merits of CQ (for example, availability of entire network of peers for the storage) and the CR (for example, the flexible handling of ACLs) approaches, at the same time, reduces the overhead of suboverlay management of a TSO to zero. We discuss some details of such a solution in [9].

Clearly, our simple threat model might not be adequate for some application scenarios, in these cases we might need to take other factors into account. Our measure leakage is closely linked to potential risks that unauthorized principals might get access to the resources. In the case of more complex threat models, the risk components might also be more difficult to capture and quantify. We do not expect that enhancing P2P overlay networks with access control possibilities will encourage companies or individuals to store very sensitive data in P2P networks, nevertheless, we believe that understanding the benefits and the risks of ACPeer systems could highly contribute to their adoption.

Acknowledgements

This work was partly supported by the European Commission (IST-35111-TEAM).

6. REFERENCES

- [1] Wuala, the social online storage. <http://www.wuala.com/>.

- [2] K. Aberer, P. Cudre-Mauroux, M. Hauswirth, and T. van Pelt. GridVine: Building internet-scale semantic overlay networks. In *International Semantic Web Conference (ISWC)*, volume 3298 of *LNCS*, pages 107–121, 2004.
- [3] 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*, pages 922–933. VLDB Endowment, 2003.
- [4] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, 2009:46–67, 2001.
- [5] 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)*, pages 585–591, Washington, DC, USA, 2005. IEEE Computer Society.
- [6] 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*, pages 95–102, Washington, DC, USA, 2002. IEEE Computer Society.
- [7] 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*, pages 9–9, Berkeley, CA, USA, 2003. USENIX Association.
- [8] 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*, pages 41–48, New York, NY, USA, 2006. ACM.
- [9] R. Narendula. Towards ACPeer: An Access Control aware Structured P2P System, technical report, lsir-report-2008-009. Technical report, Swiss Federal Institute of Technology (EPFL), 2009.
- [10] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., Second edition, 1996.
- [11] A. Singh. *Secure Management of Networked Storage Services: Models and Techniques*. PhD thesis, College of Computing, Georgia Institute of Technology, 2007.
- [12] R. Steinmetz and K. Wehrle. Peer-to-peer systems and applications. *Lecture Notes in Computer Science*, 2005.
- [13] 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*, page 74, Washington, DC, USA, 2006. IEEE Computer Society.
- [14] 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)*, pages 835–840, Washington, DC, USA, 2005. IEEE Computer Society.