Protecting Data Privacy in Structured P2P Networks

Mohamed Jawad, Patricia Serrano-Alvarado, Patrick Valduriez

Protecting Data Privacy in Structured P2P Networks

Mohamed Jawad[1], Patricia Serrano Alvarado[1], and Patrick Valduriez[2]

[1] LINA, University of Nantes
[2] INRIA and LINA, University of Nantes

Abstract. P2P systems are increasingly used for efficient, scalable data sharing. Popular applications focus on massive file sharing. However, advanced applications such as online communities (e.g., medical or research communities) need to share private or sensitive data. Currently, in P2P systems, untrusted peers can easily violate data privacy by using data for malicious purposes (e.g., fraudulence, profiling). To prevent such behavior, the well accepted Hippocratic database principle states that data owners should specify the purpose for which their data will be collected. In this paper, we apply such principles as well as reputation techniques to support purpose and trust in structured P2P systems. Hippocratic databases enforce purpose-based privacy while reputation techniques guarantee trust. We propose a P2P data privacy model which combines the Hippocratic principles and the trust notions. We also present the algorithms of PriServ, a DHT-based P2P privacy service which supports this model and prevents data privacy violation. We show, in a performance evaluation, that PriServ introduces a small overhead.

1 Introduction

Peer-to-Peer (P2P) systems provide efficient solutions for distributed data sharing which can scale up to very large amounts of data and numbers of users. Online peer-to-peer (P2P) communities such as professional ones (e.g., medical or research) are becoming popular due to increasing needs on data sharing. In such communities, P2P environments offer valuable characteristics (e.g., scalability, distribution, autonomy) but limited guarantees concerning data privacy. They can be considered as hostile because data, that can be sensitive or confidential, can be accessed by everyone (by potentially untrustworthy peers) and used for everything (e.g., for marketing, profiling, fraudulence or for activities against the owners preferences or ethics).

Data privacy is the right of individuals to determine for themselves *when*, *how* and *to what* extent information about them is communicated to others [14]. It has been treated by many organizations and legislations which have defined well accepted principles. According to OECD[1], data privacy should consider: collec-

[1] Organization for Economic Co-operation and Development. One of the world's largest and most reliable source of comparable statistics, on economic and social data. http://www.oecd.org/

A. Hameurlain and A M. Tjoa (Eds.): Globe 2009, LNCS 5697, pp. 85–98, 2009.

tion limitation, purpose specification, use limitation, data quality, security safeguards, openness, individual participation, and accountability. From these principles we underline *purpose* specification which states that data owners should be able to specify the purpose (data access objective) for which their data will be collected and used. Several solutions that follow the OECD guidelines have been proposed. A major solution is Hippocratic databases [2,9] where purpose-based access control is enforced by using privacy metadata, i.e. privacy policies and privacy authorizations stored in tables. A privacy policy defines for each attribute, tuple or table the access purpose, the potential users and retention period while privacy authorization defines which purposes each user is authorized to use.

In addition to purpose-based data privacy, to prevent data misuse, it is necessary to trust users. Reputation techniques verify the trustworthiness of peers by assigning them *trust levels* [7,10,13]. A trust level is an assessment of the probability that a peer will not cheat.

Motivations. In the context of P2P systems, few solutions for data privacy have been proposed. They focus on a small part of the general problem of data privacy, e.g. *anonymity* of uploaders/downloaders, *linkability* (correlation between uploaders and downloaders), *content deniability*, data encryption and authenticity [3,8]. However, the problem of data privacy violation due to data disclosure to malicious peers is not addressed.

As a motivating example, consider a collaborative application where a community of researchers, doctors, students and patients focus on the evolution of cardiovascular diseases (e.g. heart attacks, atherosclerosis, etc.). In such application, doctors share selected patients' records, researchers share last research results and this information is considered as sensitive. In order to control disclosure without violating privacy, data access should respect the privacy preferences defined by concerned users, for instance:

- A researcher may allow reading access on her research results to doctors for *diagnosing* and to students for *analyzing*.
- A doctor may allow writing access on her diagnosis to researchers for *adding comments*.

In this P2P application, sharing data (i.e., medical records, research results) based on privacy preferences is a challenge. Purposes defined by users (e.g. diagnosing, analyzing, etc.) should be respected. In addition, data should not be shared equally by all users. It is necessary to consider the concept of trust among users. For instance, doctors need to trust researchers to share their private data with them. Currently, P2P systems do not take into account privacy preferences. In this context, an efficient P2P purpose-based privacy service with trust control is needed.

Contributions. This paper has two main contributions.

- We propose a P2P data privacy model in which we combine several concepts related to P2P data privacy. We use this model as a basis to enforce users privacy preferences.

– We propose PriServ[2], a DHT privacy service which, based on the proposed model, prevents privacy violation by limiting malicious data access. For that, we use purpose-based access control and trust techniques. In PriServ, we consider that private data are stored and managed by their owners. The system manages only data references. To our knowledge, PriServ is the first proposition that introduces data access based on purposes in P2P systems. The performance evaluation of our approach through simulation shows that the overhead introduced by PriServ is small.

Next, Section 2 discusses related work. Section 3 presents our P2P data privacy model. Section 4 presents PriServ our privacy service. Section 5 describes performance evaluation. Section 6 shows current work. Section 7 concludes.

2 Related Work

The first work that uses purposes in data access is Hippocratic databases [2]. Inspired by the Hippocratic Oath and guided by privacy regulations, authors propose ten principles that should be preserved, namely, purpose specification, consent of the donor, limited collection, limited use, limited disclosure, limited retention, accuracy, safety, openness and compliance. Subsequent works have proposed solutions for Hippocratic databases. In [9], the goal is to enforce privacy policies within existing applications. In [1], the authors address the problem of how current relational DBMS can be transformed into their privacy-preserving equivalents. In this paper, compared to those works which enforce purpose-based disclosure control in a centralized relational datastore, we apply the Hippocratic database principles to P2P networks.

Many P2P systems propose access control services. OceanStore [8] provides efficient location of data. Protected data are encrypted and access control is based on two types of restrictions: *reader* and *writer* restrictions. Compared to OceanStore, our work improves private data access by adding the notion of purpose. Freenet [3] is a distributed storage system which focuses on privacy and security issues. It uses anonymous communications. Messages are not send directly from sender to recipient, thus, uploader and downloader anonymity is preserved. Besides, all stored files are encrypted so that a node can deny the knowledge of the content. In this work, we deal with data privacy protection and we do not address peer anonymity nor content deniability.

Other works propose trust-based management systems [7,10,13]. In [13], a considerable number of trust models and algorithms have been presented to tackle the problem of decentralized trust management. In [7], authors employ a shared global history of peer interactions to identify potential malicious owners. In [10], a peer A asks for trust levels to a limited number of peers called friends. Friends are trustworthy peers from the point of view of A. A calculates a mean value from collected trust levels. PriServ uses [10] because it generates a low

[2] In [6], our first ideas were proposed.

trust level searching cost. Unlike [10] where the trustworthiness of data owners is verified, in our work the trustworthiness of data requesters is verified.

Our privacy service marries reputation-based trust and purpose-based access control. Trust-based systems do not take into account the notion of purpose. This makes PriServ different from the existent trust management systems.

3 P2P Data Privacy Model

This section presents our P2P data privacy model.

Peer Types. In our model, we can have three distinguished types of peers:

- **Requester.** A peer that requests data in the system.
- **Owner.** A peer that provides data to the system. It owns the data it shares.
- **Reference manager.** A peer that stores meta-information as data references, owner identifiers, etc.

Privacy Policy. Each data owner can have its own privacy preferences which are reflected in privacy policies. A privacy policy can include:

- **Users.** Peers who have the right to access data. A user can be an individual or a group.
- **Access rights.** Authorizations that determine what a peer can do with data. We use three basic types of access: read, write and disclose (delegate).
- **Access purposes.** Objectives for which peers can access data.
- **Conditions.** Conditions under which data can be accessed. This may concern data values, for example age>10.
- **Obligations.** Obligations which a user must accomplish after the data access, for example a researcher Ri should return research results after using the record of patient x.
- **Retention time.** A time to limit the retention of the data, for example, a researcher should destroy the record of patient x after 6 months.
- **Minimal trust levels.** Minimal trust levels[3]which requester peers should have in order to gain access to data.

Trust. It is a fuzzy concept where trusted peers are supposed to respect privacy policies defined by data owners. Trust control uses many concepts:

- **Trust levels.** They reflect a peer reputation with respect to other peers. A peer can have different trust levels at different peers. Trust levels vary in a range of [0,1]. For instance, an *honest peer* can have a trust level in a range of [0.5, 1] and a *malicious peer* in a range of [0, 0.5]. Peers can locally register the trust levels of some peers which have interacted with them.
- **Reputation.** A peer reputation is an overall estimation of the peer behavior generally calculated from its trust levels given by peers who know it.
- **Friends.** A friend of a peer P is a peer with a high trust level from P's point of view. The number of friends held by a peer can vary from one peer to another.

[3] These levels' definition depends on the used trust model and the owner's opinion [13].

Operations. A peer can publish data, request data, or search for a trust level.

publishing(dataID, purpose). An owner uses this function to publish its data references created from its private data (*dataID*) and the corresponding access purpose (*purpose*). Publishing references allow to share private data without violating their privacy. These references allow requesters to find owners and ask them for data access. An owner has complete control of its private data which guarantees data privacy.

requesting(dataID, purpose). A requester uses this function to request data (*dataID*) for a specific purpose (*purpose*). Including the access purpose in data requests represents a contract between requesters and owners. At the same time, that makes requesters aware of their responsibilities on the data usage.

searchTrustLevel(peerID1, peerID2, nestL). A peer (*peerID1*) uses this function[4] to search the trust level of another peer (*peerID2*). The parameter *nestL* defines the nested level of searching.

Data Model. In order to respect privacy policies, we define a specific data model where privacy policies are associated with data[5].

Data Table. Each owner stores locally the data it wants to share in *data tables*. Tables 1 and 2 show two data tables. Table 1 contains the research result of researcher Ri and Table 2 contains medical records of doctor Dj.

Table 1. Data table of researcher Ri

Data table DTi			
Age	**Gender**	**Smoker**	**Risk of heart attack**
18 - 35	Female	No	7 % (rated 1)
35 - 50	Male	Yes	50 % (rated 4)
50 - 80	Female	Yes	80 % (rated 5)

Table 2. Data table of doctor Dj

Data table DTj						
PatientID (PK)	**Name**	**Country**	**Birthdate**	**Gender**	**Smoker**	**Diagnosis**
Pat1	Alex	France	1990	Male	Yes	No cardiovascular disease
Pat2	Chris	France	1973	Male	No	Cardiovascular disease (rated 2)
Pat3	Elena	Russia	1968	Female	Yes	Cardiovascular disease (rated 4)

Purpose Table. It contains information about the available purposes: purpose identifier, purpose name, purpose description, etc.

Privacy Policies Table. Each owner stores data contained in privacy policies in a table named *privacy policies table*. In this table, each line corresponds to a privacy policy which contains an id, data subject to privacy (table, column or line), access rights (read, write, disclose), allowed users, access purposes, conditions, and the required minimal trust level of allowed users. Table 3 shows the privacy policies table of doctor Dj.

[4] The use of the trust function is optional.

[5] In this paper, we use relational tables. However our model supports any type of data (XML documents, multimedia files, etc.).

Table 3. Privacy policies table of doctor Dj

Privacy policies table PPTj								
ID	Data			Access right	User	Purpose	Condition	Minimal trust level
	Table	Column	PK					
PP1	DTj	—	Pat3	r/w	Di	Updating	—	0.65
PP2	DTj	Diagnosis	Pat2	r	Pat2	Seeing diagnosis	—	0.5
PP3	DTj	—	—	r/w/d	Dj	Monitoring	—	0.9
PP4	DTj	Birthdate	—	r	Researchers	Research on cardiovascular disease	Birthdate < 2000	0.6

4 PriServ

In this section, we present PriServ, a service which, based on the privacy model of the previous section, prevents privacy violation in DHT-based systems.

4.1 Design Choices

In this section, we present PriServ main design choices.

DHT. All DHT systems (e.g. Chord [12], Pastry [11], etc.) support a distributed lookup protocol that efficiently locates the peer that stores a particular data item. Data location is based on associating a *key* with each data item, and storing the key/data item pair at the peer to which the key maps. A DHT maps a key k to a peer P called *responsible for k with respect to a hash function h.* Every peer has the IP address of log N peers in its finger table. DHT provides two basic operations, each incurring $O(logN)$ messages.

- $put(k, data)$ stores a key k and its associated data in the DHT.
- $get(k)$ retrieves the data associated with k in the DHT.

All DHT systems can be used in PriServ. We choose *Chord* for its efficiency and simplicity.

Data keys. In PriServ, peer identifiers are chosen by hashing the peer IP address and data keys are calculated by hashing the pair (*dataID, purpose*). *dataID* is a unique data identifier and *purpose* is the data access purpose[6]. *dataID* should not contain private data (e.g. patient name, country, etc.) but only meta-information (table name, column, etc.) (see Table 4). Because keys include access purposes, requesting data is always made for a defined purpose. This allows to enhance purpose-based access control.

Required Tables. In PriServ, each reference manager maintains locally a *reference table* which contains keys and corresponding owner identifiers. Each owner maintains locally a *trust table* and a *private table.* A trust table contains the trust level of some peers in the system. A private table shows the mapping of the requested keys with corresponding privacy policies.

[6] [4] has analyzed how to manage schemas and process simple queries efficiently in a flexible P2P environment. Thus, we consider that all peers accessing the same data are capable of schema mapping and that peers allowed to access particular data are informed of their allowed purposes. Thus, requester peers are able to produce keys.

Table 4. Private table of the doctor Dj

Private table			
DataID	**Purpose**	**Key**	**Privacy policy**
DTj.Pat3	Updating	21	PP1
DTj	Monitoring	71	PP3
DTj.Birthdate	Researching on cardiovascular disease	83	PP4
DTj.Pat3	Having second diagnosis	96	PP5

4.2 PriServ Algorithms

PriServ implements the P2P data privacy model proposed in Section 3. In particular, it focuses on the publishing, requesting and searchTrustLevel operations.

Publishing algorithm. In PriServ, when a peer enters the system, it uses the *put(key, ownerID)* function to publish the key/ownerID pair of the data it shares. The owner hashes the data identifier and the access purpose parameters of the $publishing(dataID, purpose)$ function to produce the data key. This allows to avoid requesting data for invalid purposes because keys corresponding to invalid access purposes will not be available in the system. Unlike the traditional DHT $put(key, data)$ function, instead of publishing shared data ($data$), the data provider identifier is published. Thus data are not published in the system. Only data providers control their data sharing.

Requesting algorithm. (Figure 1). Data requesting is done in two steps. First, the requester hashes the data identifier and the access purpose parameters of the $requesting(dataID, purpose)$ function to produce the data key. It searches for the reference manager in order to get the owner identifier bu using the $get(key)$ function. Second, the requester contacts the owner to retrieve data. The owner verifies locally the privacy policy corresponding to the data key. In particular, the owner verifies the trustworthiness of the requester.

Trust level searching algorithm. If a peer (ID) has the trust level of the requester ($requesterID$) in its trust table, this level is returned directly and the peer (ID) does not have to contact other peers. If a peer does not have the requester trust level, PriServ defines three methods for searching the requester trust level in the system:

– *With-friends algorithm* (Figure 2). Each peer has at least one friend, the owner asks its friends for the trust level of the requester. Each received trust level (RTL) is weighted with the trust level (FTL) of the sending friend. The final trust level is computed from the received trust levels. Searching for the requester trust level is recursive[7]. If a friend does not have the requested

[7] In order to assure that a peer will not be contacted twice, we consider the trust level searching as a tree in which the owner is the root and the depth is equal to the maximum depth of searching. Contacted peers form the tree nodes. If a peer already exists in the branch where it should be added, it will not be contacted a second time.

```
0: requesting(dataID,purpose)
1: begin
2:   ownerID is initialized to null;
3:   requestedData is initialized to null;
4:   key = hashFunction(dataID,purpose);
5:   ownerID = DHT.get(key);
6:   if (ownerID not null) do
7:     requestedData = retrieve(requesterID,
                    ownerID, key);
8:   return requestedData
9: end;

10: retrieve(requesterID, ownerID, key)
11: begin
12:  requestedData is initialized to null;
13:  nestL is intitialized to 0;
14:  if(requesterID has the rigth to
      access data correspondant to key) do
15:    requesterTrustL = searchTrustLevel
           (ownerID,requesterID,nestL);
16:  if(requesterTrustL is higher than
         minTrustL)
17:    requestedData = data correspondant
                              to key;
18:  return requestedData
19: end;
```

Fig. 1. Requesting algorithm

```
0: searchTrustLevel(ID,requesterID,nestL)
1: begin
2:  requesterTrustLevel is initialized to 0;
3:  if(nestL has reached MaxDepth)
4:   if(trustL of requesterID in trustTable)
5:    requesterTrustL=trustL of requesterID;
6:   else
7:    return -1;
8:  else
9:  if(trustL of requesterID in trustTable)
10:   requesterTrustL=trustL of requesterID;
11: else
12:   nestL is incremented;
13:   NbPeersContacted is initialized to 0;
14:   for each friend
15:     FTL = trustL of friendID;
16:     RTL = searchTrustLevel(friendID,
              requesterID,nestL);
17:     if (RTL != -1)
18:      FTL*RTL is added to requesterTrustL;
19:      NbPeersContacted is incremented;
20:   requesterTrustL = requesterTrustL /
                NbPeerscontacted;
21: return requesterTrustL
22:end;
```

Fig. 2. SearchTrustLevel: with-friends

trust level it asks for it to its friends and the number of nested levels (*nestL*) is incremented. Recursion is limited by a predefined number of iterations (MaxDepth). The maximum number of contacted friends can also be limited to a predefined number.

- *Without-friends algorithm.* Each peer does not have friends. The algorithm will proceed in the same way as the with-friends algorithm. However, instead of contacting friends, an owner will contact the peers in its finger table ($O(LogN)$ peers).
- *With-or-without-friends algorithm.* Each peer may have friends or not. In this case, if an owner has some friends, it uses the with-friends algorithm, otherwise it uses the without-friends algorithm.

Figure 3 illustrates the data requesting algorithm and the with-friends algorithm to search for trust levels. P18 requests data corresponding to key 21 from P25. P25 returns P36 which is the owner peer of 21. Then P18 contacts P36. P36 contacts its friends (P54) to find the trust level of P18. P54 does not have the trust level of P18 in its trust table so it contacts also its friends (P25). P25 has the trust level and sends it to P54. P54 updates its trust table and sends the trust level of P18 to P36. As it is higher than the level required in PP1 (i.e., 0.65) the access is granted.

To summarize, PriServ allows to prevent malicious access based on privacy policies and requester trust levels. A contract between requesters and owners is established by including access purposes to data requests. Owners control the

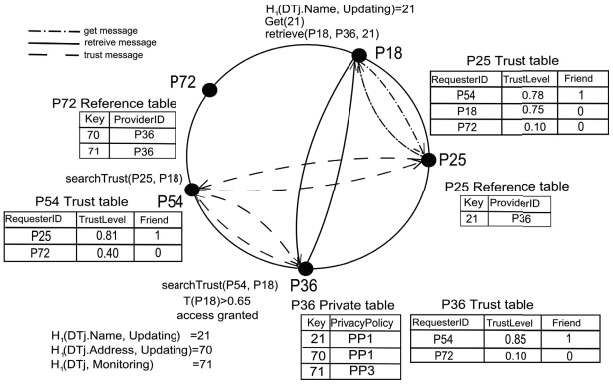


Fig. 3. Data requesting: with-friends algorithm

access to their private data and requesters are aware of their responsibilities on the data usage.

4.3 Cost Analysis

In this section, we analyze the costs of the previous algorithms in terms of number of messages. We do not analyze the join/leave cost which is the same of Chord with the advantage that data transfer of a leaving/joining peer is lighter in PriServ because only data references are stored in the system.

Publishing, data requesting and retrieving costs are in $O(logN)$ as explained in [6]. However trust level searching cost (C_{STL}) is different in function of the trust searching algorithm:

- *With-friends algorithm.* This cost depends on the number of friends (NF) and the maximum depth of the nested search (MaxDepth).

$$C_{STL_{WF}} = \sum_{i=1}^{MaxDepth} NF^i = O(NF^{MaxDepth})$$

- *Without-friends algorithm.* This cost depends on the number of fingers which is log N and the maximum depth of the nested search (MaxDepth).

$$C_{STL_{WOF}} = \sum_{i=1}^{MaxDepth} (logN)^i = O((logN)^{MaxDepth})$$

- *With-or-without-friends algorithm.* This cost depends on the number of friends (NF), the number of fingers which is log N and the maximum depth of the nested search (MaxDepth).

$$C_{STL_{WWF}} = O((max(logN, NF))^{MaxDepth})$$

The trust level searching cost C_{STL} can be one of the three costs $C_{STL_{WF}}$, $C_{STL_{WOF}}$ or $C_{STL_{WWF}}$. Note that if NF > log N, $C_{STL_{WWF}}$ is equal to $C_{STL_{WF}}$, else it is equal to $C_{STL_{WOF}}$. In all cases $C_{STL_{WWF}}$ can be used for C_{STL}:

$$C_{STL} = O((max(logN, NF))^{MaxDepth})$$

To summerize,

$$C_{publishing} = O(nbData * nbPurpose * logN)$$
$$C_{Requesting} = C_{Request} + C_{Retrieve} + C_{STL}$$
$$= 2O(logN) + O((max(logN, NF))^{MaxDepth}) = O((max(logN, NF))^{MaxDepth})$$

Compared to the traditional DHT functions, $C_{request}$ and $C_{retrieve}$ costs do not introduce communication overhead in term of number of messages. However, C_{STL} increases the data requesting cost due to the nested search. Next section shows how this cost can be reduced and stabilized to a minimum cost.

5 Performance Evaluation

In [6], we have shown by simulation that publishing and data requesting costs are in $O(logN)$ and that having ten access purposes per datum (which is realistic) PriServ is scalable. This section evaluates the performance of the trust level searching cost and the stabilization of the trust level searching cost.

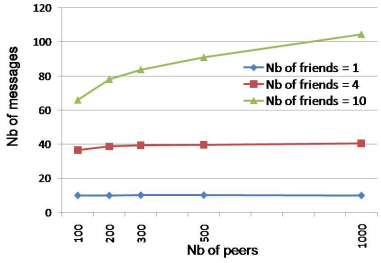
For the simulation, we use SimJava [5]. We simulate the Chord protocol with some modifications in the *put*() and *get*() functions. The parameters of the simulation are shown in Table 5. In our tests, we consider N peers with a number of data keys equal to the number of data multiplied by the number of purposes. Data and peer keys are selected randomly between 0 and 2^n. In our simulation, we set n to 11 which corresponds to 2^{11} peers. This number of users is largely sufficient for collaborative applications like medical research.

Trust Level Searching Cost. We measure the trust level searching cost (in number of messages) versus the number of peers for the three algorithms.

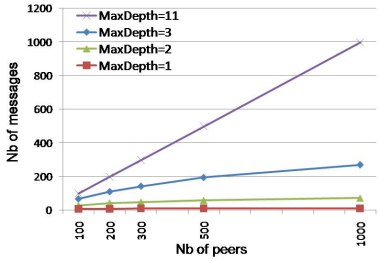
With-friends algorithm. We consider that peers have the same number of friends (*NF*) and the maximum depth (*MaxDepth*) is set to 11 (the highest maximum depth we allow in the simulation). Figure 4.a illustrates 3 measures where we consider 1, 4 and 10 friends. Those measures are slightly different of the cost model where the cost is $O(NF^{MaxDepth})$ thanks to our tree-based optimization and because the probability to contact twice a peer in a system of 100 peers is higher than in a system of 1000 peers. That is why in Figure 4.a, the trust level searching cost increases with the number of peers. We observe that for a small number of friends the trust level searching costs depends only on the number of friends as predicted by our cost model.

Without-friends algorithm. Figure 4.b illustrates 4 measures where the maximum depth of searching varies between 1, 2, 3 and 11. We recall that the number of contacted peers is $log(N)$. Thus, the trust level searching costs is logarithmic for small values of depth. This cost increases with the maximum depth of searching as predicted by our cost model.

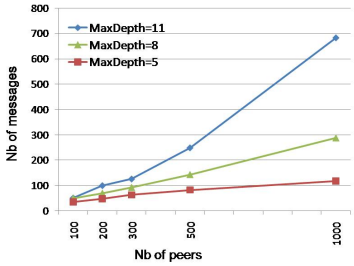
With-or-without-friends algorithm. We consider that the probability that a peer has friends in our simulation is 0.9. Figure 4.c illustrates 3 measures where the maximum depth of searching varies between 5, 8 and 11. We observe that the



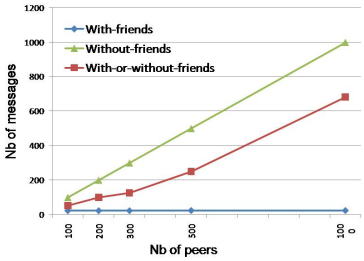
a. With-friends algorithm



b. Without-friends algorithm



c. With-or-without-friends algorithm



d. Comparison of the three algorithms

Fig. 4. Trust level searching costs

trust level searching cost is rather logarithmic for small values of depth. This cost increases with the maximum depth as predicted by our cost model[8].

Comparison. Figure 4.d compares the three algorithms seen above. We consider a number of friends equal to 2 and a maximum depth equal to 11. As predicted before, the with-friends case introduces the smallest cost while the without-friends case introduces the highest. However, intuitively, the probability to find the trust level is higher in the without-friends algorithm than in the with-friends algorithm. This is due to the fact that the number of contacted peers is higher in the without-friends algorithm, which increases the probability to find the trust level. We estimate that the with-or-without-friends algorithm is the most optimized because it is a tradeoff between the probability to find the requester trust level and the trust level searching cost.

Stabilization of the Trust Level Searching Cost. We now focus on the number of messages used to search the trust level of a requesting peer versus the number of its requests. Here we consider the three algorithms of trust (see Figure 5). We observe that the number of messages decreases and stabilizes after a number of searches. This is due to the fact that the more a peer requests for data, the more it gets known by the peers in the system.

[8] We do not measure in this case the trust level searching cost versus the number of friends which will be the same as the with-friends case.

Table 5. Table of parameters

Simulation parameters		
Variable	**Description**	**default**
n	Number of bits in the key/peer	11
N	Number of peers	2^{11}
FC	Number of friends	2
MaxDepth	Maximum depth of trust searching	11

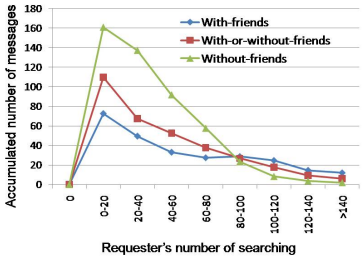


Fig. 5. Stabilization of the trust level searching cost

When peers ask for a trust level, answers are returned in the requesting order and the trust tables are updated with the missing trust level. Thus, the trust tables evolve with the number of searches. After a while, these tables stabilize. Thus, the number of messages for searching trust levels is reduced to a stable value. This value is not null because of the dynamicity of peers[9].

We also observe in Figure 5, that the trust level searching cost in the without-friends algorithm stabilizes first. This is due to the fact that a larger number of peers are contacted in this algorithm. The with-or-without-friends algorithm comes in second place, and the with-friends algorithm comes last. As we have seen in the comparison of the three algorithms, we find again that the with-or-without-friends algorithm is the most optimized because it is a tradeoff between the time to stabilization and the trust level searching cost.

6 Current Work

Storing data on owner peers gives a maximal guaranty of privacy protection. However, this hypothesis may affect availability if the owner peer fails or leaves.

To improve availability, we are working on extending PriServ functionalities where owners will choose to store locally their data or to distribute them on the system. Because distribution depends on the DHT, owners could see their private data stored on untrusted peers. To overcome this problem, before distribution, data will be encrypted and decryption keys will be stored and duplicated by owners. This will guarantee that a) private data are protected from malicious server peers, and b) an owner is able to control the access to its data since a requester peer should contact him to obtain decryption keys. In addition, owners will be able to recover their data from local storage failures.

Another extension of PriServ focuses on verification to detect misbehaviors of malicious peers. The idea is to be able of auditing the storage space dedicated to PriServ by monitoring data distribution and data access.

[9] In our simulation, we consider that the number of peers joining the system is equal to those leaving the system. Thus, there are always new peers which do not know the requester trust level.

7 Conclusion

In this paper, we addressed the problem of protecting data privacy in structured P2P systems. We apply the Hippocratic database principle to enforce purpose-based privacy. We also use reputation-based trust management in order to verify the trustworthiness of requester peers. Our solution is a P2P data privacy model which combines Hippocratic principles and trust notions. We also proposed the algorithms of PriServ, a DHT-based P2P privacy service which supports this model and prevents data privacy violation. The performance evaluation of our solution through simulation shows that the overhead introduced by PriServ is small.

References

1. Agrawal, R., Bird, P., Grandison, T., Kiernan, J., Logan, S., Rjaibi, W.: Extending Relational Database Systems to Automatically Enforce Privacy Policies. In: IEEE Conference on Data Engineering, ICDE (2005)
2. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Hippocratic Databases. In: Very Large Databases, VLDB (2002)
3. Clarke, I., Miller, S.G., Hong, T.W., Sandberg, O., Wiley, B.: Protecting Free Expression Online with Freenet. IEEE Internet Computing 6(1) (2002)
4. Furtado, P.: Schemas and Queries over P2P. In: Andersen, K.V., Debenham, J., Wagner, R. (eds.) DEXA 2005. LNCS, vol. 3588, pp. 808–817. Springer, Heidelberg (2005)
5. Howell, F., McNab, R.: Simjava: a Discrete Event Simulation Library for Java. In: Society for Computer Simulation, SCS (1998)
6. Jawad, M., Serrano-Alvarado, P., Valduriez, P.: Design of PriServ, A Privacy Service for DHTs. In: International Workshop on Privacy and Anonymity in the Information Society (PAIS), collocated with EDBT (2008)
7. Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The Eigentrust Algorithm for Reputation Management in P2P networks. In: ACM World Wide Web Conference, WWW (2003)
8. Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S.E., Eaton, P.R., Geels, D., Gummadi, R., Rhea, S.C., Weatherspoon, H., Weimer, W., Wells, C., Zhao, B.Y.: OceanStore: An Architecture for Global-Scale Persistent Storage. In: Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS) (2000)
9. LeFevre, K., Agrawal, R., Ercegovac, V., Ramakrishnan, R., Xu, Y., DeWitt, D.J.: Limiting Disclosure in Hippocratic Databases. In: Very Large Databases, VLDB (2004)
10. Marti, S., Garcia-Molina, H.: Limited Reputation Sharing in P2P Systems. In: ACM Conference on Electronic Commerce, EC (2004)
11. Rowstron, A.I.T., Druschel, P.: Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In: ACM/IFIP/USENIX Middleware Conference (2001)

12. Stoica, I., Morris, R., Karger, D.R., Kaashoek, M.F., Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In: ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, SIGCOMM (2001)
13. Suryanarayana, G., Taylor, R.N.: A Survey of Trust Management and Resource Discovery Technologies in Peer-to-Peer Applications. Technical report, UCI Institute for Software Research, university of California, Irvine (2004)
14. Westin, A.: Privacy and Freedom. Atheneum, New York (1967)