Design of PriServ, A Privacy Service for DHTs

Mohamed Jawad, Patricia Serrano-Alvarado, Patrick Valduriez

Design of PriServ, A Privacy Service for DHTs

Mohamed Jawad
LINA

Patricia Serrano Alvarado
LINA

Patrick Valduriez
INRIA and LINA

University of Nantes
{Mohamed.Jawad,Patricia.Serrano-Alvarado}@univ-nantes.fr
Patrick.Valduriez@inria.fr

ABSTRACT

By decentralizing control, P2P systems provide efficient, scalable data sharing. However, when sharing data for different *purposes* (e.g., billing, purchase, shipping, etc.), data privacy can be easily violated by untrustworthy peers wich may use data for other purposes (e.g., marketing, fraudulence, profiling, etc.). A basic principle of data privacy is *purpose* specification which states that data providers should be able to specify the purpose for which their data will be collected and used. In the context of P2P systems, decentralized control makes it hard to enforce purpose-based privacy. And the major problem of data disclosure is not addressed. Hippocratic databases provide mechanisms for enforcing *purpose-based* disclosure control within a corporation datastore. In this paper, we apply the Hippocratic database principles to P2P systems to enforce purpose-based privacy. We focus on Distributed Hash Tables (DHTs), because they provide strong guarantees in terms of access performance. We propose PriServ, a privacy service which prevents privacy violation by prohibiting malicious data access. The performance evaluation of our approach through simulation shows that the overhead introduced by PriServ is small.

1. INTRODUCTION

Peer-to-Peer (P2P) systems provide an efficient solution for distributed data sharing. By decentralizing control, they can scale up to very large amounts of data and numbers of users. P2P systems have been mainly used for file sharing. Recently, the needs to support advanced applications led to *content sharing* where shared data are semantically richer than files (e.g., relational tables, XML documents, etc.). By significantly increasing the types of shared data and the ways to access them, content-sharing in P2P also rises the risk of misuse. Data providers may see their data used for marketing, illegal competition, or activities against their 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 [11]. The main principles underlying data privacy, according to OECD[1], are: collection limitation, purpose specification, use limitation, data quality, security safeguards, openness, individual participation, and accountability. Several solutions that follow the OECD guidelines have been proposed. A major solution is Hippocratic databases which can enforce purpose-based disclosure control in a relational datastore [1, 6]. This is achieved 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 usage purpose(s), the potential users and retention period while privacy authorization defines which purposes each user is authorized to use.

1.1 Motivations

In the context of P2P systems, decentralized control makes it hard to deal with data privacy. A few solutions have been proposed but focus on a small part of the general problem, e.g. *anonymity* of uploaders/downloaders, *linkability* (correlation between uploaders and downloaders), *content deniability*, data encryption and authenticity [3, 5]. However, the major problem of data disclosure is not addressed. As a motivating example, let us consider a Master degree in Computer Science. Several teachers may participate in the same course. Students are evaluated and graded by each teacher. The final grade of students is computed based on the teachers' grades. One of the teachers is responsible for a course and gathers all grades. Students may ask for their grades or their ranking, e.g. according to the final grade, to one particular teacher, etc.

We consider a P2P system where each teacher, represented by a peer, manages its grades locally. Teachers should be able to specify their privacy preferences for data access. For instance:

- a teacher may allow reading access to her colleagues participating in the same course *for particular rankings* of students or to her students *for their rankings or grades*,
- the responsible of a course may allow updating access to the teachers participating in the course but only *for adding their grades*; or she may allow reading access to the responsible of the degree *for knowing the average grade*.

In this P2P application, sharing data (i.e., grades) based on privacy preferences is a challenge. Purposes defined by teachers, such as adding grades, ranking wrt a specific tea-

[1]http://www.oecd.org/.

cher, etc. should be respected. Data should not be shared indistinctly with all users. Currently, distributing and requesting data in P2P systems do not take into account privacy preferences. So controlling data sharing for specific purposes is not possible in such systems without adding new services.

In this context, an efficient P2P purpose-based privacy service is needed.

1.2 Contributions

In this paper, we apply the Hippocratic database principles to P2P systems to enforce privacy. In particular, we propose to include access purposes in data distribution and data requesting to prevent violating data privacy, and to verify trust levels of clients to prevent sharing data with suspicious or malicious peers.

We focus on Distributed Hash Tables (DHTs), because they provide strong guarantees in terms of access performance.

This paper has two main contributions.

- First, we propose a privacy model for DHTs. In such model we define data models where privacy policies are integrated.
- Second, we propose PriServ, a privacy service which prevents privacy violation by prohibiting malicious data access and use. For that, we use purpose-based access control and trust notions.

The performance evaluation of our approach through simulation shows that the overhead introduced by PriServ is reasonable.

We present our privacy model for DHTs in section 2. PriServ is presented in section 3. We describe our performance evaluation in section 4. We discuss related work in section 5 and conclude in section 6.

2. PRIVACY MODEL FOR DHT

Privacy protection can be divided into two phases: *prevention* and *verification*. The prevention phase consists in preventing privacy violation by prohibiting malicious data access and use. The verification phase consists in detecting privacy violation and taking actions against malicious users. In this paper, we focus on the first phase.

In the next, we define the DHT model (section 2.1), privacy policies (section 2.2) and the data model (section 2.3).

2.1 DHT Model

All DHTs 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 using a hash function h. We call P the *responsible for k wrt h*. The responsible for k *wrt* h may be different at different times because of peer joins and leaves.

Many DHTs have been proposed, e.g. Chord [9], Pastry [8], etc. To generalize, DHT provides two basic operations, each incurring $O(logN)$ messages.

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

All DHTs can be used in this work. We chose Chord for its efficiency and simplicity. In Chord, peers maintain information about $O(logN)$ other peers in a *finger table* and resolve lookups via ***O(logN)*** messages to other peers. A finger table entry includes both the Chord identifier and the IP address (and port number) of the relevant peer. A consistent hash function assigns to each peer and key an m-bit $identifier$ using a base hash function. A peer identifier is chosen by hashing the peer IP address. A key identifier is based on data values that can be a data identifier, an address, etc. All peers are ordered in a circle modulo 2^m. Key k is assigned to the first peer whose identifier is equal to or follows k in the identifier space. Chord provides data distribution and searching using only ***O(logN)*** messages.

2.2 Privacy Policies

Each data provider can have its own privacy preferences (every teacher has her privacy preferences). These preferences can include the users that have the right to access data, what kind of access a user has (read/write), whether users have the right to disclose shared data, the purpose of data access, the minimal trust level of the user, etc. Those data privacy preferences are reflected in privacy policies.

We consider that each data provider is responsible for defining and maintaining her privacy policies in an independent way.

All concepts in the privacy policy model are important. We focus on the access purpose and the trust level elements which are poorly addressed by access control systems. The *access purpose* states the data access objective. The *trust level* is an assessment of the probability that a peer will not cheat.

2.3 Data Model

In order to respect privacy policies, in particular, the notion of purpose and trust, we define a specific data model where privacy policies are associated with data.

We consider that each peer stores locally the data it wants to share in relational tables which we call *data tables*. Data contained in privacy policies are stored in a table named *privacy policies table*. Following our motivating example, Table 1 shows a data table which contains the student grades of the course responsible Rj. Table 2 shows the privacy policies table of the course responsible Rj. In this table, one tuple corresponds to one privacy policy. Each policy contains an id, data subject to privacy (table, column or tuple), access rights (read, write, delete), allowed users, access purposes, conditions (if they exist), and the required minimal trust level of allowed users.

3. PRISERV

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

Section 3.1 gives our design choices, section 3.2 algorithms to distribute and search data and section 3.3 a cost analysis of these algorithms.

3.1 Design Choices

Although our design is general enough to be used in different DHTs, we use Chord to illustrate how PriServ works. Our main design choices concern the introduction of purpose into data keys, the use of trust levels, and the definition of new tables to deal with privacy.

Data table DTj				
Student (PK)	**Grade1**	**Grade2**	**Exam grad**	**Average**
S1	15.0	13.0	13.0	13.66
S3	9.5	10.0	8.0	9.00

Table 1: Data table of the course responsible Rj

Privacy policies table j								
ID	**Data**			**Access right**	**User**	**Purpose**	**Condition**	**Minimal trust level**
	Table	**Column**	**PK**					
PP1	DTj	Grade1	—	r/w	Pi	Add grades	—	0.65
PP3	DTj	—	—	r/w/d	Rj	Monitoring	—	0.9
PP4	DTj	Average		r	Students	Ranking	Average>10	0.6
PP5	DTj	—	S3	r	S3	Grade details	—	0.7

Table 2: Privacy policies table of the course responsible Rj

Private table			
Data_id	**Purpose**	**Key**	**Privacy policy**
DTj.Grade1	Add grades	21	PP1
DTj	Monitoring	71	PP3
DTj.Average	Ranking	83	PP4

Table 3: Private table of the course responsible Rj

3.1.1 Data Keys

In PriServ, like in Chord, peer identifiers are chosen by hashing the peer IP address. Concerning the data key, to enforce data privacy, we propose to hash the pair (*data_id, purpose*). *data_id* is a unique data identifier and *purpose* is the data access purpose. Thus, the same data with different access purposes have different keys.

Because keys contain the notion of purposes, searching data is always made for a defined purpose. This allows to enhance access control because to construct data keys, data requesters have to include the purpose for which the data are accessed. We assume that data requesters know the purposes for which data are accessed.

3.1.2 Trust Levels

PriServ uses trust levels to make the final decision of sharing or not data. The trust level reflects a peer reputation wrt other peers. A peer can have different trust levels at different peers.

The peer reputation influences its trust level. Peers which are suspicious have lower trust level than peers considered as honest. Peers can have locally the trust levels of some well known peers or peers which have interacted with them. If a peer does not have a particular trust level it can ask for this to other peers. These peers are called friends. A *friend* is a peer considered as honest from the peer P's point of view. The number of friends can vary from one peer to another.

3.1.3 Required Tables

We consider that storing data in the system may affect data privacy because data providers do not control the access to their data. In PriServ, data are locally stored at the provider peer. Only keys and the corresponding provider identifier are distributed in the system. Thus, in PriServ peers can play the following roles.

- **Requester**: peer that makes a data request.
- **Responsible**: peer responsible for key/provider pair.
- **Provider**: peer that shares data with requesters. Normally, it owns the data.

Each responsible peer maintains locally a *reference table* which contains keys and corresponding provider identifiers. Provider peers use *private tables* (e.g., Table 3) where keys and privacy policies are associated. This table facilitates the mapping of the requested keys with corresponding privacy policies. Each provider also maintains a local *trust table*. This table contains the trust value of some peers in the system.

All these tables are locally stored at each peer in a secure database management system.

3.2 PriServ Algorithms

The PriServ service uses two main algorithms, one for key distribution and another for data requesting.

3.2.1 Key Distribution Algorithm

In PriServ, when a peer enters the system, it uses the *put (key, provider_id)* function to distribute the key/provider_id pair of the data it shares. The key is produced by hashing the data identifier and the access purpose (*hashFunction(data_id, purpose)*). The provider_id is the key that identifies the entering peer. A peer already connected to the system can share data by using the $hashFunction$ and the *put* functions.

Unlike the traditional DHT $put(key, data)$ function, instead of distributing shared data ($data$), the data provider identifier is distributed. Thus data are not distributed in the system to enhance data privacy. Only data providers control their data sharing.

3.2.2 Data Requesting Algorithm

When a peer requests a data item, it produces the corresponding key and searches for the peer responsible for that key in the system. The peer responsible returns the provider identifier to the requester. The requester then addresses the request to the provider peer. Finally, the provider verifies its privacy policies and the trust level of the requester. If the requester access rights are compatible with those defined in the provider privacy preferences, the provider shares the data with the requester.

The requesting algorithm has three main functions.

1. $search(data\_id, purpose)$. The requester hashes the data identifier and the access purpose to produce the key. It searches for the responsible of the key in order to get the provider identifier. This is done by using the $get()$ function.
2. $retrieve(requester\_id, provider\_id, key)$. The provider verifies locally the privacy policy of the key with the

$verifyPP(requester\_id, key)$ function. If this verification is ok, the provider searches for the trust level of the requester with the $searchTrust()$ function. If the trust level is equal to or higher than the one specified in the corresponding privacy policy, data are accessed.

3. $searchTrust(requester\_id)$. In order to find the trust level of requesting peers, the provider searches the trust value locally in its trust table. If the value does not exist, the provider asks its friends for it. Each received trust level is weighted with the trust level of the sending friend. Then from all the trust levels is computed an average. This searching is recursive. If a friend does not have the requested trust level it asks for it to its friends. Recursion can be limited by a predefined number of iterations.

Due to space restriction, we do not show an example of this algorithm. To summarize, PriServ guarantees limited collection of private data since a provider controls the access to its private data. Thus PriServ allows to prevent malicious access based on privacy policies.

3.3 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.

3.3.1 Key Distribution Cost

Using the DHT, we need $O(logN)$ messages to distribute each key. In PriServ, the number of keys is equal to the number of entries of the private table (ept). Thus, the distribution cost is:

$$C_{dist} = \sum_{i=1}^{ept} O(logN) = O(ept * logN)$$

The maximum value of ept is equal to the number of shared data ($nbData$) multiplied by the number of purposes ($nbPurpose$), i.e., at worst, each data item is shared for all purposes:

$$CMax_{dist} = O(nbData * nbPurpose * logN)$$

We can see that the number of purposes affects the key distribution cost. Previous studies have shown that considering 10 purposes allows to cover a large number of applications [7]. Used with 10 purposes (by data item), PriServ incurs a small overhead. Overall, the key distribution cost remains logarithmic.

3.3.2 Data Requesting Cost

This cost is obtained by adding the costs of the three main functions used in the requesting algorithm.

1. $search(data\_id, purpose)$. It uses the traditional DHT $get()$ function which requires $O(logN)$ messages.

$$C_{search} = O(logN)$$

2. $retrieve(requester_{id}, provider_{id}, key)$. The requester contacts the provider using the DHT so the cost is $O(logN)$ messages. By using the IP address, communication can be direct and the cost is one message.

$$C_{Retreive} = O(logN) \text{ or } 1$$

3. $searchTrust(requester_{id})$. In this function, the provider sends a message to each of its friends which in turn do the same in a nested search. This cost depends on the number of friends (NF) and the depth of the nested search (D). We consider that each message sent has a response so the cost is two times NF. Let NF_i be the number of friends at the i^{th} search.

$$\begin{aligned} C_{searchTrust} &= \sum_{i=1}^{D} 2 * NF_i \\ &= O(D * N) \\ &= O(N) \end{aligned}$$

Thus, the data requesting cost is:

$$\begin{aligned} C_{request} &= C_{search} + C_{retrieve} + C_{searchTrust} \\ &= 2 * O(logN) + O(N) \\ &= O(logN) + O(N) \\ &= O(N) \end{aligned}$$

To summarize, in terms of communication messages and compared to the traditional DHT functions, the C_{search} and $C_{retrieve}$ costs are not modified. However, $C_{searchTrust}$ increases the data requesting cost due to the nested search. The next section shows how this cost can be reduced. However this cost is always linear.

Simulation parameters	
Variable	**Description**
n	number of bits in the key/peer
N	number of peers
FC	number of friends
nbPurpose	number of purposes
nbData	number of data

Table 4: Table of parameters

4. PERFORMANCE EVALUATION

This section evaluates the performance of PriServ by simulation. We focus on the key distribution (Section 4.1) and data requesting (Section 4.2) costs, and the stabilization of the searching cost of the requester trust value (Section 4.3).

For the simulation, we use SimJava [4]. We simulate the Chord protocol with some modifications in the $put()$ and $get()$ functions. The parameters of the simulation are shown in Table 4.

4.1 Key Distribution Cost

We measure the number of messages for distributing one data item (nbData is equal to 1) in function of the number of peers. Figure 1 illustrates 5 measures where the number of purposes (nbPurpose) goes from 1 to 10. Note that with only 1 purpose, we have a system without the real notion of purpose. We can observe that the distribution cost is logarithmic and increases with the number of purposes. We recall that having 10 purposes for each data item is an extreme case.

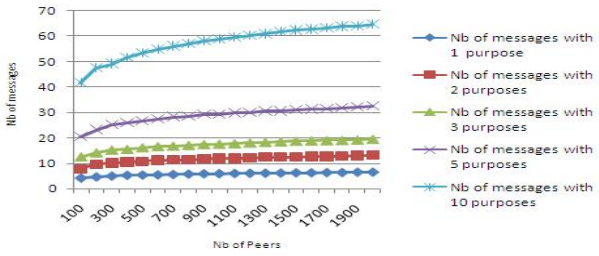

Figure 1: Key distribution cost

4.2 Data Requesting Cost

We measure the number of messages for requesting one data item in function of the number of peers. Figure 2 shows two costs. The first cost is the number of messages to get the provider identifier. The second cost is the sum of the first cost and the number of messages to get the private data. We observe that the requesting costs are logarithmic as predicted by our cost model (see Section 3.3.2).

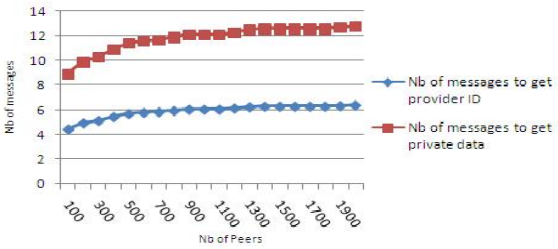

Figure 2: Searching cost

4.3 Stabilization of the Trust Searching Cost

We now focus on the number of messages used to search the trust value of a requesting peer in function of the number of its requests. Here we consider 4 friends per peer. The result is illustrated in Figure 3. 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.

The trust tables evolve with the number of searches. After a while, these tables stabilize. Thus, the number of messages for searching trust values reduces to a stable value which is not null due to dynamicity of peers.

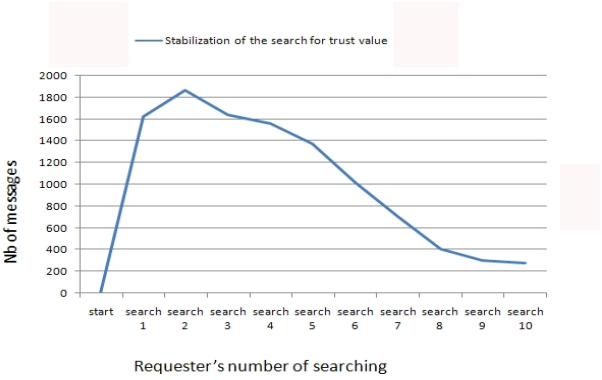

Figure 3: Stabilization of the trust searching cost

5. RELATED WORK

The first work that uses purposes in data access is [1]. Inspired by the Hippocratic Oath and guided by privacy regulations, the authors propose ten principles that should be preserved in Hippocratic databases, 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 [2], the authors propose query modification techniques and Role-Based Access Control (RBAC) to ensure data privacy based on purposes.

In this paper, we use the Hippocratic database principles [1], mainly, access purposes. We propose to extend P2P functionalities with a service that uses privacy policies tables where data and purposes are linked.

To protect data, OceanStore [5] and Piazza [10] use public/private keys. PriServ does not use these keys because they are insufficient to protect data privacy wrt providers preferences. Once data are decrypted, they could be used for different purposes and their privacy can be violated.

6. CONCLUSION

In this paper, we addressed the problem of data privacy in DHTs. We proposed PriServ, a privacy service which prevents privacy violation by prohibiting malicious data access. PriServ uses the notion of purposes in data access control and the notion of trust. The performance evaluation of our approach through simulation shows that the overhead introduced by PriServ is small. However, we demonstrated that the more a peer requests data, the more it gets known by the peers in the system and its trust level searching cost is reduced.

7. REFERENCES

[1] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic Databases. In *Very Large Databases (VLDB)*, 2002.

[2] J.-W. Byun, E. Bertino, and N. Li. Purpose Based Access Control of Complex Data for Privacy Protection. In *ACM Symposium on Access Control Models and Technologies (SACMAT)*, 2005.

[3] I. Clarke, S. G. Miller, T. W. Hong, O. Sandberg, and B. Wiley. Protecting Free Expression Online with Freenet. *IEEE Internet Computing*, 6(1), 2002.

[4] F. Howell and R. McNab. Simjava: a Discrete Event Simulation Library for Java. In *Society for Computer Simulation (SCS)*, 1998.

[5] J. Kubiatowicz, D. Bindel, Y. Chen, S. E. Czerwinski, P. R. Eaton, D. Geels, R. Gummadi, S. C. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Y. Zhao. OceanStore: An Architecture for Global-Scale Persistent Storage. In *Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2000.

[6] K. LeFevre, R. Agrawal, V. Ercegovac, R. Ramakrishnan, Y. Xu, and D. J. DeWitt. Limiting Disclosure in Hippocratic Databases. In *Very Large Databases (VLDB)*, 2004.

[7] 1.0 P3P Purposes of Data Collection Elements. http://p3pwriter.com/LRN_041.asp.

[8] A. I. T. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In *ACM/IFIP/USENIX Middleware Conference*, 2001.

[9] I. Stoica, R. Morris, D. R. Karger, M. F. Kaashoek, and H. Balakrishnan. 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.

[10] I. Tatarinov, Z. G. Ives, J. Madhavan, A. Y. Halevy, D. Suciu, N. N. Dalvi, X. Dong, Y. Kadiyska, G. Miklau, and P. Mork. The Piazza Peer Data Management Project. *ACM SIGMOD Record*, 32(3), 2003.

[11] A. Westin. Privacy and Freedom. In *Atheneum*, New York, 1967.