Peer-to-Peer Private Information Retrieval

Josep Domingo-Ferrer and Maria Bras-Amorós

Universitat Rovira i Virgili
UNESCO Chair in Data Privacy
Department of Computer Engineering and Mathematics
Av. Països Catalans 26, E-43007 Tarragona, Catalonia
{josep.domingo,maria.bras}@urv.cat

Abstract. Private information retrieval (PIR) is normally modeled as a game between two players: a user and a database. The user wants to retrieve some item from the database without the latter learning which item. Most current PIR protocols are ill-suited to provide PIR from a search engine or large database: i) their computational complexity is linear in the size of the database; ii) they assume active cooperation by the database server in the PIR protocol. If the database cannot be assumed to cooperate, a peer-to-peer user community is a natural alternative to achieve some query anonymity: a user submits a query on behalf of another user in the community. A peer-to-peer PIR system is described in this paper which relies on an underlying combinatorial structure to reduce the required key material and increase availability.

Keywords: Privacy in statistical databases, private information retrieval, combinatorial designs.

1 Introduction

In private information retrieval (PIR), a user wants to retrieve an item from a database or search engine without the latter learning which item the user is interested in. PIR was invented in 1995 by Chor, Goldreich, Kushilevitz and Sudan [3,4] with the assumption that there are at least two copies of the same database, which do not communicate with each other. In the same paper, Chor *et al.* showed that single-database PIR (that is, with a single copy) is infeasible in the information-theoretic sense. However, two years later, Kushilevitz and Ostrovsky [12] presented a method for constructing single-database PIR based on the algebraic properties of the Goldwasser-Micali public-key encryption scheme [6]. Subsequent developments in PIR are surveyed in [15].

In the PIR literature the database is usually modeled as a vector. The user wishes to retrieve the value of the i-th component of the vector while keeping the index i hidden from the database. Thus, it is assumed that the user knows the physical address of the sought item, which might be too strong an assumption in many practical situations. Keyword PIR [2,12] is a more flexible form of PIR: the user can submit a query consisting of a keyword and no modification in the structure of the database is needed.

J. Domingo-Ferrer and Y. Saygin (Eds.): PSD 2008, LNCS 5262, pp. 315–323, 2008.

We claim that PIR protocols proposed so far have two fundamental shortcomings which hinder their practical deployment:

1. The database is assumed to contain n items and PIR protocols attempt to guarantee maximum privacy, that is, maximum server uncertainty on the index i of the record retrieved by the user. Thus, the computational complexity of such PIR protocols is $O(n)$, as proven in [3,4]. Intuitively, all records in the database must be "touched"; otherwise, the server could rule out some of the records when trying to discover i. For large databases, an $O(n)$ computational cost is unaffordable [1].
2. It is assumed that the database server cooperates in the PIR protocol. However, it is the user who is interested in her own privacy, whereas the motivation for the database server is dubious. Actually, PIR is likely to be unattractive to most companies running queryable databases, as it limits their profiling ability. This probably explains why no real instances of PIR-enabled databases can be mentioned.

If one wishes to run PIR against a search engine, there is another shortcoming beyond the lack of server cooperation: the database cannot be modeled as a vector in which the user can be assumed to know the physical location of the keyword sought. Even keyword PIR does not really fit, as it still assumes a mapping between individual keywords and physical addresses (in fact, each keyword is used as an alias of a physical address). A search engine allowing only searches of individual keywords stored in this way would be much more limited than real engines like Google or Yahoo.

In view of the above, in [11] a system is proposed in which a user masks her target query by ORing it with $k-1$ fake queries and then submits the masked query to a search engine or large database which does not need to cooperate (in fact, it does not even need to know that the user is trying to protect her privacy). Rather than total privacy, this system provides a sort of k-privacy, in that it cloaks the target query within k queries. This system works fine but assumes that the frequencies of keywords and phrases that can appear in a query are known and available: for maximum privacy, the frequencies of the target and the fake queries should be similar.

1.1 Contribution and Plan of This Paper

We present a peer-to-peer private information retrieval system. It has the same practical philosophy of [11]; however, rather than cloaking a query in a set of queries, the system described here cloaks a user in a peer-to-peer user community, because a user submits queries on behalf of her peers and conversely. This approach certainly requires the availability of peers (not needed in [11]) but it has the advantage of not requiring knowledge of the frequencies of all possible keywords and phrases that can be queried.

The new scheme uses a type of combinatorial design called configuration to increase service availability and reduce the number of required keys (see [16,13] for

background on designs and configurations). The use of configurations in cryptographic key management is not new (*e.g.* see [13]), but their use in private information retrieval is.

Section 2 presents configurations. Section 3 describes the proposed peer-to-peer PIR protocol. Section 4 assesses the performance and the privacy offered by the protocol. Finally, Section 5 sketches conclusions and future work.

2 (v, b, r, k)-Configurations

We first define a combinatorial design and then a configuration as a special type of design.

Definition 1 (design). *A* design *is a pair $(X, \mathcal{A})$, where X is a set of points and $\mathcal{A}$ is a finite set of subsets of X, called blocks. The degree of a point $x \in X$ is the number of blocks containing x. The rank of $(X, \mathcal{A})$ is the size of the largest block.*

A design is said to be *regular* if all points have the same degree, say r. A design is said to be *uniform* if all blocks have the same size, say k (in which case the design is uniform of rank k). In the next definition we used the notations in [16,13].

Definition 2 ($(\mathbf{v}, \mathbf{b}, \mathbf{r}, \mathbf{k})$-1-design). *A (v, b, r, k)-1-design is a regular and uniform design with $|X| = v$, $|\mathcal{A}| = b$, degree r and rank k.*

A (v, b, r, k)-1-design corresponds to a bipartite semiregular graph with $v+b$ vertices and degrees r and k. A necessary and sufficient condition for the existence of a (v, b, r, k)-1-design is that

$$bk = vr. \tag{1}$$

Definition 3 ($(\mathbf{v}, \mathbf{b}, \mathbf{r}, \mathbf{k})$-configuration). *A (v, b, r, k)-configuration is a (v, b, r, k)-1-design where any two distinct blocks intersect in zero or one point.*

A (v, b, r, k)-configuration corresponds to a bipartite semiregular graph with $v+b$ vertices, degrees r and k, and girth strictly larger than 4. Configurations and their history have been largely studied by Harald Gropp in [7,8,9,10].

The following lemma quantifies the "connectivity" between blocks in a configuration.

Lemma 1. *In a (v, b, r, k)-configuration the number of blocks intersecting any specific block is $k(r-1)$.*

Proof. Consider a (v, b, r, k)-configuration $(X, \mathcal{A})$ and fix a block $A_i \in \mathcal{A}$. For any $x \in A_i$ define

$$\mathcal{B}_x = \{A_j \in \mathcal{A} : x \in A_j\} \setminus \{A_i\}$$

Clearly, $|\mathcal{B}_x| = r - 1$ for all $x \in A_i$. On the other hand, the sets $\mathcal{B}_x$ $(x \in A_i)$ are disjoint. Thus, the number of blocks intersecting A_i can be computed as

$$\left| \bigcup_{x \in A_i} \mathcal{B}_x \right| = \sum_{x \in A_i} |\mathcal{B}_x| = k(r-1). \qquad \square$$

A necessary condition for the existence of a (v, b, r, k)-configuration is $v \geq r(k-1)+1$ [5]. Yet, this condition may not be sufficient.

Finding configurations and even determining if a configuration with a given set of parameters exists is not trivial. One can use the next greedy algorithm to find a (v, b, r, k)-configuration if it exists.

We think of $\mathcal{A}$ as a list of b lists. The ith list in $\mathcal{A}$ correspond to the points in X contained in the ith block. We initialize all lists in $\mathcal{A}$ to empty lists. We use a pair of indices (i, j), where i and j respectively indicate which list in $\mathcal{A}$ and what position in the list we are dealing with. We start with $(i, j) = (1, 1)$ that is, we start by enumerating the first point of the first block.

Then we proceed by appending to the ith list in $\mathcal{A}$ a $(j+1)$th point, while the ith list has less than k elements or by appending a first point to the $(i+1)$th list. When this is not possible because there is no possible point to append satisfying the requirements of a configuration then backtracking is used (the last assignements of points to blocks are deleted in reverse chronological order until one is found that can be changed in a way compatible with the configuration structure).

It is helpful to use a second list of lists corresponding to the v points in X, with the lth list indicating what blocks contain the lth point. It can be used for verifying if a given point at a given position of a block in $\mathcal{A}$ is possible.

The algorithm is the following one:

while$(0 < i \leq b)$

$$test = \begin{cases} 0 & \text{if } A_{i,j} = NULL \text{ and } j = 1 \\ A_{i,j-1} + 1 & \text{if } A_{i,j} = NULL \text{ and } j \neq 1 \\ A_{i,j} + 1 & \text{if } A_{i,j} \neq NULL \end{cases}$$

while($test \leq v - k + j$ and appending $test$ to A_i does not lead to a configuration)

$test = test + 1$

if$(test = v - k + j + 1)$

$A_{i,j} = NULL$

$$(i, j) = \begin{cases} (i-1, k) & \text{if } j = 1 \\ (i, j-1) & \text{if } j \neq 1 \end{cases}$$

BACKTRACK

else

$A_{i,j} = test$

$$(i, j) = \begin{cases} (i, j+1) & \text{if } j \neq k \\ (i+1, 1) & \text{if } j = k \end{cases}$$

if$(i = 0)$

output $\emptyset$

if$(i = b + 1)$

output A.

We refer the reader to [10] for results on existence and particular examples of configurations.

3 A Peer-to-Peer PIR Protocol

Consider a peer-to-peer (P2P) community consisting of b users. Assume a dealer who creates a key pool in the following way:

1. The dealer creates v keys and distributes them into b blocks of size k each according to a (v, b, r, k)-configuration.
2. The dealer confidentially sends one block of k keys to each user (no two users get the same block). E.g., if each user has got a public-private key pair, confidentiality can be achieved by sending the block of keys encrypted under the user's public key. Let A_i be the block assigned to user u_i, for $i = 1$ to b.
3. The dealer erases the v keys from its storage. If a trusted device such as a smart card is used as a dealer, it can be assumed that keys are forgotten by the dealer after distribution.

A variant of the above initialization process is to allow the users to send to the dealer their preferences about which other users they would like to share keys with. The dealer could take this input into account to the extent possible when assigning blocks of keys to users.

At the end of the process, by Lemma 1 the block of keys of each user intersects $k(r-1)$ other users' blocks. Consider now a storage pool consisting of v memory sectors, each corresponding to one key in the key pool.

Note 1. In the above set-up process, users do not need to know each other's identity. When deciding whether identities are to remain pseudonymous or not, one should carefully ponder whether the increased mutual trust derived from mutual knowledge compensates the loss of privacy of users in front of the rest of users in the P2P community. We will henceforth assume user pseudonymity. See Section 4.2 below for further discussion on user privacy.

A protocol for peer-to-peer PIR among the b users is specified next. The keys distributed to the users are used to key a symmetric cipher (*e.g.* see [14]). Also, in what follows we assume that plaintext queries and answers to queries can be distinguished from garbage by a user decrypting them; some kind of redundancy (*e.g* a cyclic redundancy check) can be appended to the query or the query answer to facilitate this distinction.

Protocol 1 (P2P PIR query submission)

1. *In order to submit a query q_i to a database or search engine, user u_i randomly selects one of the k keys in her block. Let x_{ij} be the selected key and $U_j^i = \{u_{j1}^i, \cdots, u_{j(r-1)}^i\}$ be the set of $r-1$ users with whom u_i shares x_{ij} according to the configuration used for key distribution. (Note that the sets $U_1^i, \cdots, U_k^i$ are disjoint due to the configuration structure.)*
2. *u_i reads the memory sector m_{ij} corresponding to key x_{ij} and decrypts it under x_{ij}. Five cases can arise depending on the outcome of decryption:*

(a) *The outcome is garbage. In this case, u_i encrypts q_i using a symmetric cipher keyed by x_{ij} and records the encrypted query in sector m_{ij}.*
(b) *The outcome is a query q_j issued by some user in U_j^i, who expects u_i to submit it on her behalf. In this case, u_i submits q_j to the database/search engine and records in sector m_{ij} the answer obtained after encrypting it under key x_{ij}. Thereafter, u_i goes back to Step 1 to select a new key and obtain assistance in submitting q_i to the database/search engine from someone in the group of $r-1$ users sharing the new key with u_i.*
(c) *The outcome is the answer to a previous query q'_j issued by some user in U_j^i and previously submitted by u_i to the database/search engine on behalf of that user. Since this answer has not yet been read by the user in U_j^i (a user is assumed to erase the query answer when she reads it), u_i goes back to Step 1 to select a new key and obtain assistance in the submission of her own query q_i.*
(d) *The outcome is a query q'_i previously issued by u_i, who expects some user in U_j^i to submit it on u_i's behalf. Since there is a previous query pending to be serviced by some user in U_j^i, u_i goes back to Step 1 to select a new key and obtain assistance with the submission of her new query q_i.*
(e) *The outcome is the answer to a previous query q'_i issued by u_i and previously submitted by some user in U_j^i to the database/search engine on behalf of u_i. In this case, u_i reads the answer, then encrypts her new query under key x_{ij} and finally records the encrypted query in sector m_{ij}.*

It can be seen that Protocol 1 will iterate until u_i can have her query submitted to the database/search engine by some other user. If a user does not have queries to submit and never runs Protocol 1, she does not contact the database; if the number of users contacting the database is very small (*e.g.* only two) there are problems: i) the database may infer who is submitting what query, and ii) the delay until a query answer can be collected can be too long. To prevent this, we require that all users run Protocol 1 at regular time intervals, whether or not they wish to submit actual queries (they can submit fake queries if necessary).

After submitting a query q_i, user u_i follows the protocol below to collect the answer to q_i:

Protocol 2 (P2P PIR query answer collection)

1. *If x_{ij} was the key selected to submit q_i, u_i keeps reading and decrypting m_{ij} at regular time intervals until either the answer to q_i is found (and erased from m_{ij}) or a timeout occurs.*
2. *If there was a timeout, u_i calls Protocol 1 to select a new key and find some other user who can assist her with the submission of q_i.*

4 Performance and Privacy

We examine in this section the influence of the configuration parameters k and r on performance and privacy. The other two parameters do not need discussion:

b is the (fixed) number of users in the P2P community and v is the number of keys and depends on k, r and b according to Equation (1).

4.1 Performance

First we deal with performance in terms of required keys and required storage. The proposed set-up process based on a (v, b, r, k)-configuration is compared with the trivial case in which every user shares a different key with every other user (complete connection graph). It turns out that performance improvement is controlled by parameter r.

Lemma 2. *If $r > 2$ it holds that:*

- *the number of keys and memory sectors required using a (v, b, r, k)-configuration is less than the number of keys and memory sectors required in the case of a complete graph;*
- *the overall number of keys stored by the users with a (v, b, r, k)-configuration is less than in the case of a complete graph.*

Proof. With a complete graph among the b users, the number of required keys and memory sectors is $b(b-1)/2$. Each user stores $b-1$ keys, so that the overall number of keys stored by the users is $b(b-1)$.

With configurations, the number of required keys and memory sectors is $v = bk/r$ (Equation (1)). The overall number of keys stored by the users is bk. Now, from Lemma 1 it follows that $k(r-1) \leq b-1$ (the number of blocks intersecting a specific block cannot be greater than $b-1$); thus

$$\frac{bk}{r} \leq \frac{b(b-1)}{r(r-1).}$$

So for $r > 2$ there is a reduction in the number of required keys and memory sectors with respect to the complete graph case. Similarly, since $bk \leq b(b-1)/(r-1)$, for $r > 2$ there is a reduction in the overall number of keys stored by the users. □

In addition to storage, another performance metric is how long does it take for u_i to get her query submitted and answered. Clearly, the greater the number r with whom u_i shares the selected key x_{ij}, the shorter is the expected waiting time.

Therefore, performance improves as r increases.

4.2 Privacy

If a good symmetric cipher is used for encryption, the encrypted contents stored in any memory sector are indistinguishable from garbage (see [14] for a review of the properties of the output of a symmetric cipher). Thus to an intruder not in $\{u_i\} \cup U_j^i$ the content of sector m_{ij} is indistinguishable from garbage; therefore,

such an intruder does not gain any information on the queries submitted nor the query answers received by users in $\{u_i\} \cup U_j^i$.

Within U_j^i, the $r-1$ users do not know in principle to which other user in $\{u_i\} \cup U_j^i$ do the queries and query answers correspond. From this remark and those in the performance section above, one might be tempted to take r as large as possible: a single key shared by all b users (that is, $r = b$ and $k = v = 1$). Yet this does not look like a good solution because then any user can see any query or query answer, which does not seem very secure nor private: even if users are pseudonymous, successive queries by the same u_i are likely to be linkable, with the subsequent profiling and re-identification risk for u_i (*e.g.* a way to link u_i's queries is through her IP address when u_i writes her queries or reads her query answers).

It seems better for u_i to limit (pseudonymous) visibility of her query and its answer to those parties strictly needed: the database/search engine and a set of users just large enough so that the expected waiting time to get the query answer is not too long. Indeed, if u_i can select x_{ij} among $k > 1$ different keys at Step 1 of Protocol 1, where each key is shared by a disjoint set of users (see proof of Lemma 1), users in U_j^i only see on average 1 out of k queries issued by u_i (and 1 out of k query answers received by u_i). Therefore, the risk that a user in U_j^i can profile and thereby re-identify u_i decreases as k increases.

Finally, let us examine the privacy of user u_i in front of the database or search engine. The queries issued by u_i are submitted by the $k(r-1)$ users with whom u_i shares keys. In fact, each u_j in that group of $k(r-1)$ users submits on average a fraction $1/(k(r-1))$ of the queries issued by u_i. But u_j may also submit other queries corresponding to other users different from u_i with whom u_j shares a key. Therefore the query profile of u_i is diffused among the $k(r-1)$ users with whom u_i shares a key and confused among the other queries submitted by those users.

In summary, the greater r, the better is performance; the greater k, the greater is privacy in front of the other users; the greater $k(r-1)$, the greater is privacy in front of the database/search engine.

5 Conclusion

So far in the literature, practical PIR protocols that can be run against an uncooperative search engine or database aim at cloaking the user query among a finite number of fake queries. We have introduced a new paradigm, in which the user herself rather than the query is cloaked; indeed, the user seeks assistance by a P2P community who submit queries on her behalf. Unlike the query cloaking approach, any complex query can be submitted with our proposal and no knowledge of the frequencies of keywords and phrases is required. The level of privacy achieved is proportional to connectivity $k(r-1)$ of the P2P community.

From the combinatorial point of view, we have also contributed a construction of configurations (the structure used for key and storage management).

Acknowledgments and Disclaimer

This work was partly supported by the Spanish Government through projects CONSOLIDER INGENIO 2010 CSD2007-00004 "ARES" and TSI2007-65406-C03-01 "E-AEGIS", and by the Government of Catalonia under grant 2005 SGR 00446. The authors are with the UNESCO Chair in Data Privacy, but their views do not necessarily reflect the position of UNESCO nor commit that organization.

References

1. Beimel, A., Ishai, Y., Malkin, T.: Reducing the servers computation in private information retrieval: Pir with preprocessing. Journal of Cryptology 17, 125–151 (2004)
2. Chor, B., Gilboa, N., Naor, M.: Private information retrieval by keywords. Technical Report TR CS0917, Department of Computer Science, Technion (1997)
3. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: IEEE Symposium on Foundations of Computer Science (FOCS), pp. 41–50 (1995)
4. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. Journal of the ACM 45, 965–981 (1998)
5. Colbourn, C.J., Dinitz, J.H. (eds.): Handbook of Combinatorial Designs, 2nd edn. Chapman and Hall/CRC, London (2007)
6. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and Systems Science 28(1), 270–299 (1984)
7. Gropp, H.: On the history of configurations. In: International Symposium on Structures in Mathematical Theories, Bilbo, pp. 263–268. Euskal Herriko Unibertsitatea (1990)
8. Gropp, H.: Configurations between geometry and combinatorics. Discrete Appl. Math. 138(1-2), 79–88 (2000); Optimal discrete structures and algorithms (ODSA 2000)
9. Gropp, H.: Existence and enumeration of configurations. Bayreuth. Math. Schr. 74, 123–129 (2005)
10. Gropp, H.: Configurations. In: Colbourn, C.J., Dinitz, J.H. (eds.) Handbook of Combinatorial Designs. Chapman & Hall/CRC, Boca Raton (2007)
11. Solanas, A., Domingo-Ferrer, J., Bujalance, S.: k-private information retrieval from privacy-uncooperative queryable databases (submitted, 2008)
12. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: single database, computationally-private information retrieval. In: Proc. of the 38th Annual IEEE Symposium on Foundations of Computer Science, pp. 364–373 (1997)
13. Lee, J., Stinson, D.R.: A combinatorial approach to key predistribution for distributed sensor networks. In: Wireless Communications and Networking Conference-WCNC 2005, vol. 2, pp. 1200–1205 (2005)
14. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A. (eds.): Handbook of Applied Cryptography. CRC Press, Boca Raton (1997)
15. Ostrovsky, R., Skeith-III, W.E.: A survey of single-database pir: techniques and applications. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 393–411. Springer, Heidelberg (2007)
16. Stinson, D.R.: Combinatorial Designs: Constructions and Analysis. Springer, New York (2003)