Unobservable Surfing on the World Wide Web: Is Private Information Retrieval an alternative to the MIX based Approach?

Dogan Kesdogan[1], Mark Borning[1], Michael Schmeink[2]

[1] Lehrstuhl für Informatik IV, RWTH Aachen,
{kesdogan, borning}@informatik.rwth-aachen.de
[2] Lehr- und Forschungsg. Stochastik, RWTH Aachen,
schmeink@stochastik.rwth-aachen.de

Abstract. The technique *Private Information Retrieval* (PIR) perfectly protects a user's access pattern to a database. An attacker cannot observe (or determine) which data element is requested by a user and so cannot deduce the interest of the user. We discuss the application of PIR on the World Wide Web and compare it to the MIX approach. We demonstrate particularly that in this context the method does not provide perfect security, and we give a mathematical model for the amount of information an attacker could obtain. We provide an extension of the method under which perfect security can still be achieved.

1 Introduction

The importance of the Internet continues to grow. Today, any user can find information about nearly any topic of interest. With the increasing use of the Internet, interest in collecting information about user behavior is also increasing. For example, an Internet bookstore would like to deduce the genre that a user reads, so that it can suggest other books of this genre to promote sales. This is of course a minor example; but if one were given all the collected information about a user in the WWW, one could characterize the user and his behavior in great detail. This is the real problem of privacy, as users may object to being profiled so specifically. Therefore, the user may want to protect his interest-data. In general the protection of interest-data can be done in two ways:

1. Indirect: A number of users ($n > 1$) forms a so-called anonymity group and requests the interest-data using an anonymity protocol like e.g. a MIX- or DC-network [7, 8, 24]. Although the server is able to identify the requested interest-data, it cannot assign this data to a specific user.
2. Direct: The user requests the interest-data and additional redundant data. Therefore, the interest-data is hidden in the overall data stream. An efficient technique for this purpose has been suggested by two different groups (see [9, 11]).

All work we know on privacy-enhancement in the WWW environment focuses on the first method, i.e. using intermediary stations like MIXes [5, 14, 27, 25, etc.]. In this paper we investigate the second approach, using Private Information Retrieval (PIR), and compare our results. Our main reasons for choosing PIR include:

- Security: PIR provides perfect security (the technique itself is mostly comparable to the technique of DC-networks) and the method of previous work provide at most probabilistic security [23].
- Trust model (resp. attacker model): The MIX technique assumes that neither all of the people using the anonymity technique ((n-1)-Attack[1]) nor all of the MIXes is use are corrupt. PIR just assumes that the servers in use are not all corrupt[2].

Of course, a direct comparison of PIR and MIX is not possible. PIR provides only a "reading" function, while MIXes can both send (write) and receive (read) messages. Therefore, we will only compare the reading function of the MIX with PIR. If we wanted to both functions we would have to combine techniques.

Our work also differs from the classical PIR approach. To date, all works about PIR consider unstructured data ("flat" data) [1, 2, 3, 4, 6, 9, 10, 11, 13, 22, 30]. In our work, we investigate the application of PIR to the WWW environment, where the data structure can be modeled as a graph (structured data).

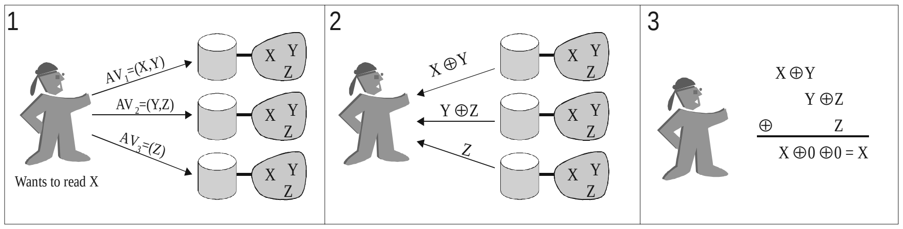

Fig. 1. Basic concept of Private Information Retrieval and message service [9, 11].

In chapter 2 we briefly describe the PIR concept. Chapter 3 discusses the problems of applying PIR to the WWW. In chapter 4 we extend PIR for hierarchical data structures. Chapter 5 discusses network architecture for Web-PIR. Finally, chapter 6 concludes this work.

2 Private Information Retrieval (PIR)

Private Information Retrieval [9] also known as *message service* [11] assures that an unbounded attacker is not able to discover the information that a user has requested (perfect security). The goal is to read exactly one datum that is stored in a memory

[1] We do not know if there is a technique providing security against the (n – 1)-Attack (see also [18, 26]).

[2] If perfect security is not envisaged then PIR does not need to trust the server [3, 6, 20].

cell of a server. To protect the privacy of the user, PIR accesses not just the single memory cell, several memory cells on several replicated servers. If not all of servers are corrupt then the method provides perfect security. The method consists of three steps:

1. The client generates mostly random vectors containing "0" and "1" and sends them end-to-end-encrypted to the appropriate servers.
2. All servers read the requested cells and superpose (XOR addition) them to one cell and send them end-to-end-encrypted to the user.
3. The user superposes all the received cells to one cell.

Figure 1 describes the idea of the method. If a customer wants to read a data item and if there are *k* PIR servers, he creates *k* – 1 random vectors. He XORs them to form the *k*th vector. In the *k*th vector, he flips the bit representing the desired data item. The customer sends one of the vectors to each server. Each server retrieves the data items corresponding to 1's in its vector, and XORs them to create one datum. The result of the XOR sum is returned to the customer who XORs all the results and obtains the requested data item.

Since the publication of the basic method, several groups have investigated this technique and extended it, e.g. the efficiency of the method has been increased (see [1, 2, 4, 10, 13, 22, 30]). In our work we will not review these extensions, since they all have the same problems with structured data. Thus, the presented problem and the suggested solution can be combined with the other improvements.

3 Lessons Learned: PIR and the World Wide Web

The data structure of the Web can be modeled as a graph [19]. In the next chapter we present an attack using the knowledge of the Web link structure. The attack is successful even though each individual page request the PIR servers is perfectly secure.

3.1 Assumptions

Using PIR the number of requests from a particular user is still observable. The attacker knows the number of links that connect one page to another, and therefore the minimum number of requests a user must make to travel that path. In this section we will discuss how to combine these two pieces of knowledge to uncover the users' interest.

For the sake of simplicity we assume the following features:

1) A particular WWW-Page will be requested only once by a user.
2) A user will request a page by mistake only from the root level, i.e. he will not mistakenly follow a path for more than one step.

One may argue that the assumptions are not general enough and therefore are not fulfilled in the most cases. However, a single example of insecurity is enough to show that PIR does not provide perfect security. In fact, these three assumption may occur often than one may believe:

- When using PIR, it is desirable that all users use their local cache, since the request cost is quite high (Feature 1).
- If a user is not familiar with a Web site, then he may surf until he finds some interesting path (Feature 2).

We believe that these assumptions are reasonable for the designer of a privacy service.

3.2 Simple Deterministic Attack

In the considered application environment it is clearer to speak of a user session rather than of visiting an abstract graph or requesting distinct pages of a Web. A user session can be seen as a process (function of time) starting from a starting node of a Web graph and reaching an arbitrary node.

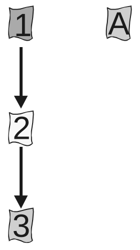

Fig. 2. Simple Web structure.

If the assumptions (1) and (2) hold, than we claim that PIR is not perfectly secure for Web browsing. Consider this example: A user requests three page from a site with the link structure as shown in Figure 2. After the first request the attacker knows that either page "*1*" or page "*A*" was requested, i.e. one element of the set {*1*, *A*}. The attacker distinguish further at this time. After the second request, however, the attacker knows that the user has requested Page "*1*" because:

- with the first request the user requested "*A*" and now "*1*", because the first request was mistaken; or
- with the first request the user requested "*1*" and now "*A* because the first request was mistaken; or
- with the first request the user requested "*1*" and now "*2*", thus a path has been chosen.

After the third request the attacker knows[3] that the user has requested Page “*1*” and Page “*2*”, and by assumption (2) now knows the user’s interest.

The example shows that a technique providing perfect protection by reading unstructured data items does not automatically lead to a technique providing perfect protection in general. In the following section generalize this result and give a mathematical model for the attacker’s information gain.

3.3 Mathematical Model for the information gain

A hierarchical data structure is a set of data elements $V := \{S_1, ..., S_n\}$ combined with a set E of links between the data elements of V. $L_0 \subseteq V$ is defined as the set of starting elements. Additionally, we assume that the edges are associated with transition probabilities. Such probabilities can be obtained by analyzing the hitlogs of a site (see [19]).

Following [29, 23] we call a request *perfectly unobservable*, if an attacker is not able to gain any information about which page is requested by observing the session. We assume that the attacker still can observe the number of requests. Thus the mathematical model follows the deterministic model (above example).

We give two models here, one for the information gained by observing one session and one for that gained observing several sessions. This distinction is useful, particularly because information can be gained by observing several sessions. Such information can be used as a general basis for a time-frequency analysis.

Single-Session Model

Using the above-mentioned transition probabilities we can define a probability distribution on the set $\underline{\mathbf{M}}$ of all possible paths that a user can take. Let g: $\underline{\mathbf{M}} \to \text{Nat}$[4]: $m \mapsto g(m) = l_m$ describe the length of a path m. The random variable M denotes the path m in $\underline{\mathbf{M}}$ explored by the current user. Let L be a random variable describing the length of the path chosen by the user. Thus, it holds that $L=g(M)$. Furthermore, let $A_l = \{m \mid m \in \underline{\mathbf{M}}, g(m) = l\}$ be the set of all paths with length l. Any opponent can count the number of accesses, and thus the length of a user's path. The ideal case would be that $P(M = m \mid L = l) = P(M = m)$ holds for all m in $\underline{\mathbf{M}}$.

Unfortunately, the attacker can rule out any path with a length different from the one observed. Thus,

$$P(M = m|L = l) = \frac{P(M = m, g(M) = l)}{P(g(M) = l)} = \frac{P(M = m, g(M) = l)}{P(M \in A_l)}$$
$$= \begin{cases} P(M = m)/P(M \in A_l), & if\ g(m) = l \\ 0, & else \end{cases}$$

[3] **Algorithm**: Write all possibilities in separate sets and take the intersection of all sets.

[4] **Nat** ≡ natural number

Let $H(M)$ denote the uncertainty measure of M and $H(M \mid L)$ the conditional uncertainty of M given L. We are interested in the information conveyed about M by L. Since L is totally dependent on M and g is a function of M, we obtain

$$\begin{aligned}
&I(M,L) = H(M) - H(M|L) = H(M) + H(L) - H(M,L)\\
&= H(M) + H(L) - H(M, g(M))\\
&= H(M) + H(L) - H(M)\\
&= H(L).
\end{aligned}$$

With a suitable choice for the logarithmic base used in the uncertainty measure, it holds that $0 \le H(L) \le 1$, where 0 denotes zero information and 1 maximum information conveyed, respectively. Thus, $H(L)$ is a measure for the mean gain in information observed during one user session. To calculate $H(L)$, we just need the probability distribution of L, given by: $P(L = l) = \sum_{m \in A_l} P(M = m)$.

Multiple-Session Model

If users are not anonymous and reveal their IP address, then opponents can gain information not only during one user session but also by observing a sequence of sessions of a single user.

For example, we can assume that the attacker can distinguish reasonable and unreasonable paths, and that a user will not choose unreasonable paths repeatedly. More generally, let $(M_1,\ldots, M_n)$ be a random vector denoting the paths chosen by a user over n sessions. An opponent will observe a random vector of path lengths $(L_1, \ldots, L_n) = (g(M_1), \ldots, g(M_n))$.

Similarly to the one-dimensional case, the information conveyed about $(M_1,\ldots, M_n)$ by $(L_1,\ldots,L_n)$ can be simplified to:

$$\begin{aligned}
&I((M_1,\ldots M_n),(L_1,\ldots L_n)) = H(M_1,\ldots M_n) - H(M_1,\ldots M_n | L_1,\ldots L_n)\\
&= H(M_1,\ldots M_n) + H(L_1,\ldots L_n) - H((M_1,\ldots M_n),(L_1,\ldots L_n))\\
&= H(M_1,\ldots M_n) + H(L_1,\ldots L_n) - H((M_1,\ldots M_n), g^*(M_1,\ldots M_n))\\
&= H(M_1,\ldots M_n) + H(L_1,\ldots L_n) - H((M_1,\ldots M_n),(g(M_1),\ldots g(M_n)))\\
&= H(M_1,\ldots M_n) + H(L_1,\ldots L_n) - H(M_1,\ldots M_n)\\
&= H(L_1,\ldots L_n)
\end{aligned}$$

Again, $H(L_1,\ldots,L_n)$ is a measure for the mean gain in information observed during a sequence of n sessions. Admittedly, the estimation of the probability distributions of $(M_1,\ldots, M_n)$ and $(L_1,\ldots,L_n)$, respectively, would lead to some practical problems.

3.4 Resulting Consequences for PIR

The direct application of PIR on structured data is unhelpful for security and for performance Normally, PIR requests data among the whole data space. Thus, in average, 50% of the whole database is requested in each random vector. But, if the assumed terms in chapter 3.1 are true, the attacker can observe the request hierarchy and exclude some of the requested pages as irrelevant because they are unrelated. Thus, there is no security gain in building a random vector over the whole data set. It is much more effective from the performance point of view (without losing any security strength) to build the vector explicitly over the particular hierarchy.

We can conclude that applying PIR on structured data reduces the anonymity set (the subset of the entire data space which the attacker must consider as possible alternatives) with the following consequences:

- To provide perfect security it has to be ensured that in each hierarchy level the size of the anonymity set is greater then 1, i.e. the Web graph must be restructured.
- From the performance point of view the random vectors should be limited to the elements of the anonymity set belonging to the hierarchy level.

4 Extensions to PIR

As described in the last section, accessing web pages in the WWW can be viewed as a user session on a hierarchical data structure. It was shown that an observer is able to determine the number of accesses, i.e. the length of the user session. If he also knows the data structure, the observer is able to exclude data items that are not reachable in this number of accesses. IN this section, we will first a method for creating hierarchy layers. Then we will describe a method to efficiently load data items into any hierarchy layer. The latter method is called *hierarchical blinded read*.

4.1 Creation of hierarchy layers

In section 3.1, we stated two assumptions defining the access to simple web structures. Within these requirements, the creation of hierarchy layers is more difficult, because a user could request $(n - 1)$ incorrect items if there are n starting items. Therefore, the hierarchy layers are enlarged as shown in the example in section 3.1.

In Fig. 3, there are four hierarchy layers. To determine these layers, we partition the web graph into three sets $L_0 := \{1, 2\}$, $L_1 := \{3, 4, 5, 6\}$ and $L_2 := \{7, 8, 9, 10, 11\}$.

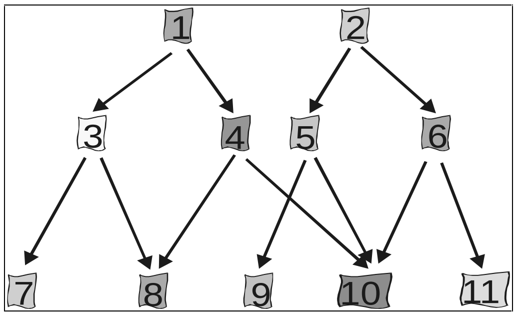

Fig. 3. Example of a web graph.

The first hierarchy layer will be the set L_0, because a user is only able to access items of this set. The second layer will be $L_0 \cup L_1$: The user may have incorrectly accessed an item of the set L_0 and is now accessing the other item; or the user has accessed an item of the set L_0 and is now accessing an item of the set L_1. The third layer will be $L_1 \cup L_2$, because the user has either accessed an item of the set L_1 in the last step and is now accessing an item of the set L_2, or he has accessed an item of the set L_0 and is now accessing an item of the set L_1. The remaining hierarchy layer is the set L_2. An example of a three-step-path is the sequence *(1, 4, 10)*; a four-step-path is the sequence *(1, 2, 5, 9)*.

In general terms: Let there be a hierarchical data structure with levels $L_0, \ldots, L_k$; and let $e \in L_i$ if e is reachable over a path of length i from the starting set. If $|L_0| = n$, then a hierarchy layer A_i, $i = 1, \ldots, k + n - 1$, is defined as $A_0 := L_0, A_i := \bigcup_{j=i-(n-1)}^{i} L_j$

For $j < 0$ and $j > k$, let $L_j = \varnothing$. Therefore, a sufficient condition for perfect secrecy is $|A_i| > 1$.

With the creation of hierarchy layers, we are able to define an algorithm to access the layers. To identify an item in a layer, the identifiers of the items are lexicographically ordered and numbered consecutively. For example, the Uniform Resource Identifier (URI) is the item's identifier in a given web structure. The URIs can be arranged in a lexicographical order.

4.2 Hierarchical Blinded Read

Having introduced a method to create hierarchy layers in hierarchical data structures that respect the requirements of section 3, we will define an algorithm in this section that realizes a blinded read access to these hierarchy layers. The algorithm is called hierarchical blinded read (HBR). The HBR algorithm is divided into two parts, the client algorithm and the server algorithm. The most important change is the introduction of a new parameter into the original algorithm, the hierarchy layer.

On the client site, the algorithm takes as input the layer l of the target item and offset o of the that item in its layer. The output is the item itself. The parameter l is also needed on the server, so the client has to send this parameter. Therefore, we extend the request vector by appending the layer *l*.

Let $n_i = |L_i|$ be the number of items in hierarchy layer L_i, *k* the number of servers, *l* the queried hierarchy layer, and *o* the offset of the target item in this layer. To simplify the calculation, we will not use binary vectors. Instead, we represent the binary vector as a natural number, where the highest order bit represents the item n_i and the lowest order bit represents the item 1 of the hierarchy layer L_i. In Fig. 3 , the number 1 ($=2^0$) represents web page "*1*" of layer 2, and the number 16 ($= 2^4$) represents web page "*5*" of the same layer.

The client algorithm has two phases, the request phase and the receive phase. In Fig. 4, the request phase of the client algorithm is shown. First, the client creates the random request vectors and randomly chooses two servers. Server *y* receives the XOR sum, and server *w* receive that value with the flipped bit representing the user's actual request. At the transmission, two function are used. *CreatePacket* creates an encrypted packet *p* that contains the request value and the hierarchy layer *l*. *SendPacket* transmits packet *p* to server *i*.

The receiving phase (Fig. 5) uses three additional functions. ReceivePacket gets a packet and stores it in r. ExtractServer gets the server's identification number from the response r, and ExtractData gets the responded data from r. Vector br is used to check whether all responses are received, and, in addition, br can be used to check if responses are sent more than once.

```
Input: k - number of servers,
       l - hierarchy layer,
       n - items in hierarchy layer,
       o - offset of wanted item in hierarchy layer
   y := random(k)
   w := random(k)
   $x_y$:= 0
   For i = 0 to k-1
      If i $\neq$ y
         x[i] := random($2^n$)
         x[y] := x[y] $\oplus$ x[i]
   x[w] := x[w] $\oplus$ $2^{o-1}$
   For i = 0 to k-1
      p := CreatePacket(l,x[i])
      SendPacket(i,p)
```

Fig. 4. Request phase of the client algorithm.

The client algorithm is almost the same as the original blinded read in [11] . The change in the request phase is the introduction of the hierarchy layer, because the layer is important in creating the request vectors and has to be transmitted to the server. In the receiving phase, the vector br prevents any server from sending multiple responses.

```
Input: k - number of server

   d := $\mathbf{0}$
   For i = 0 to k-1
      br[i] := 0
   While br $\neq$ (1, ..., 1)
      ReceivePacket(r)
      s := ExtractServer(r)
      e := ExtractData(r)
      If br[s] = 0
         d := d $\oplus$ e
         br[s] := 1
   return d
```

Fig. 5. Receive phase of the client algorithm.

If a server receives a query, he loads the corresponding items, creates their XOR sum, and sends it as a response packet to the client. Additionally, the response packet contains the unique identification number of the server.

The server algorithm (Fig. 6) uses some additional functions. *ReceivePacket* waits for a new packet and stores it in *p*; *ExtractLayer* gets the hierarchy layer *l* from the packet *p*; *ExtractRequest* gets the request value *x* from *p;* and *GetLayerSize* determines the layer's number of items *n*. After that, the corresponding items are loaded, i.e. it is checked what bits are set in the request value. After creating the XOR sum, *CreatePacket* is used to create the response packet and *SendResponse* transmits it to the client.

```
Input: s - Identification number of the server

   p := ReceivePacket
   l := ExtractLayer(p)
   x := ExtractRequest(p)
   n := GetLayerSize(l)
   e := $\mathbf{0}$
   For i = 1 to n
      If (x AND $2^{i-1}$) $\neq$ 0
         e := e $\oplus$ d[l,i]
   r := CreatePacket(s,e)
   SendResponse(r)
```

Fig. 6. Server algorithm.

The server algorithm differs from the original algorithm in [11], because the items are accessed differently, requiring the server to get the hierarchy layer with a request. Furthermore, the server transmits a unique identification number, because the client has to address the responses to the various servers.

Command	y	w	Vector x	i
y := random(3)	2	?	(?,?,?)	?
w := random(3)	2	1	(?,?,?)	?
x_y := 0	2	1	(?,?,0)	?
For i = 0 to 2. If i = y, then x_i := random (32) $x_y := x_y \oplus x_y$	2 2 2	1 1 1	(25,?,25) (25,53,44) (25,53,44)	0 1 2
$x_w := x_w \oplus 16$	2	1	(25,37,44)	?

Fig. 7. Request phase of the client

Fig. 7 shows an example execution of the request phase. Web page “5” of the Fig. is requested, i.e. the input for the algorithm is l = 2, o = 5. There are k = 3 servers and the number of pages in the hierarchy layer 2 is n2 = 6. In the loop, only the result of every run is listed. Furthermore, the request values x1, x2 and x3 are presented as a vector of length 3.

Fig. 8 shows the processing of an access at the first server. The server gets a request vector containing l = 2 and x = 25. The binary representation of x is 11001, i.e. the server has to load the pages “1”, “4” and “5”. The server creates the XOR sum of them, creates a response packet containing the sum and his identification number, and transmits the packet to the client.

In Fig. 9, the loading process of page “5” is shown. The client generates three requests and transmits them to the servers. The servers access their data repository to load the requested pages, create the XOR sum, and transmit the resulting items. The client combines the responses using the XOR function and gets the target page “5”.

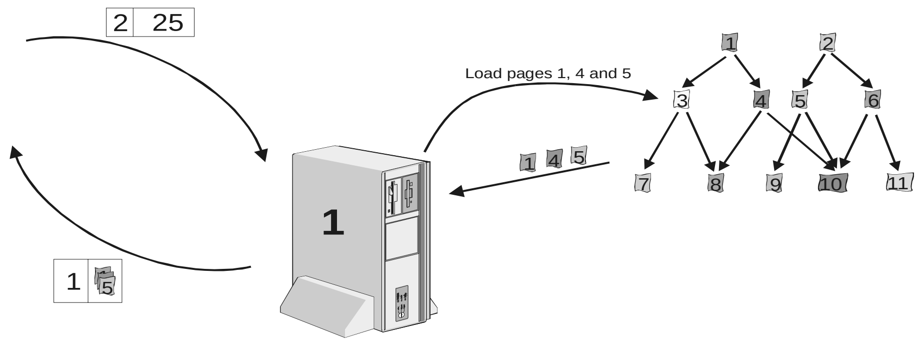

Fig. 8. Handling of on access at the server.

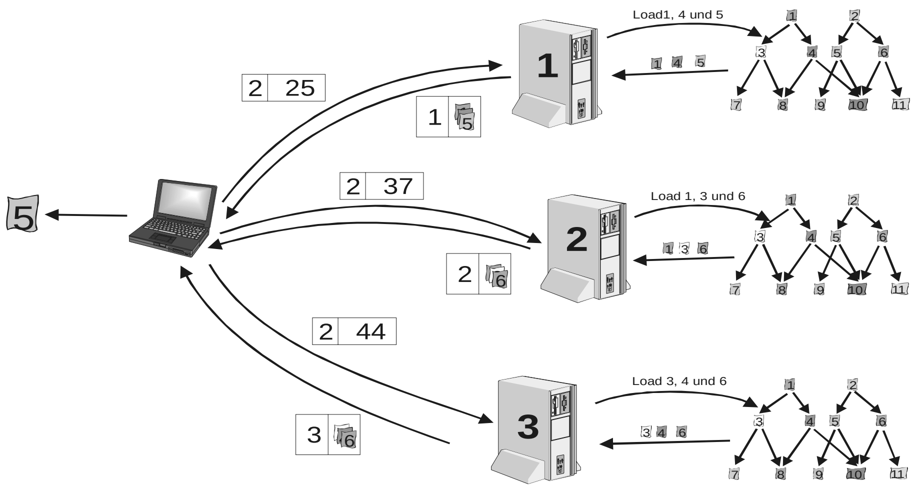

Fig. 9. Loading of a page from hierarchy layer 2.

4.3 Remarks

The HBR algorithm requires an additional parameter as input, the hierarchy layer. On the other hand, the HBR algorithm could calculate this parameter itself. Moreover, the parameter need not be given at all: If the server is able to identify a client, it can determine the number of accesses by this client and calculate the hierarchy layer itself. The administration of such an approach is very high, however, because the server would have to store the number of accesses of any identified client in an interval. Therefore, the interval has to be specified, so exactly one user session will be in it. Furthermore, the number of clients could be very large, and all of them would have to be stored. If we calculate the hierarchy layer in the client's HBR algorithm, we have to determine the interval of a user session, too. For these reasons, we have chosen an implementation in which the user himself determines of the hierarchy layer.

5 Blinded Read Networks

Loading of a data item using blinded read requires some independent servers. All servers form the so-called *Blinded Read Network*. A client has to access all servers to receive a data item. In the previous sections, we described the client and server interaction. In this section, we will describe an architecture that could be used for the blinded read.

5.1 Closed Architecture

The *closed architecture* is almost the same as the static approach of the original blinded read. Every server contains all data items that can be accessed over the blinded read network. An application area of the closed architecture is the World Wide Web. The application area covers databases that contain slowly changing data, e.g. a news archive or product catalogues of HiFi-components. The change of the database, i.e. the change, the deletion, or insertion of an item, requires propagation onto every server. Therefore, it is necessary to define synchronization times to change the database to the new version.

The structure of a blinded read server of the closed architecture is shown in Fig. 10. The blinded read server has an interface any client can connect to. In its trusted area, a database contains the available data, i.e. the database is stored at the same computer as the blinded read server or it can be accessed very efficient over a local network.

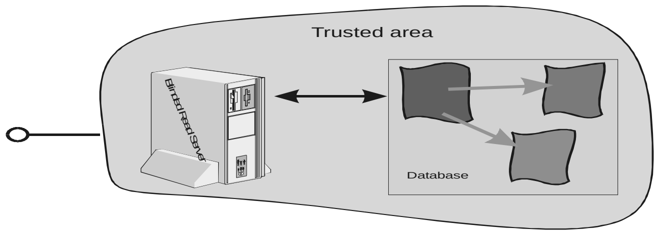

Fig. 10. Closed Architecture.

A client needs an address table to access the data items. The blinded read server will create this table and will transmit it to any client who requests it. The table further contains the synchronization times, so the client knows how long this table will be valid, and when he must request a new table.

The advantage of the closed architecture is the direct and fast access to the data items. Furthermore, the architecture has the same security properties as the blinded read, so no additional protection is necessary. However, a close architecture has limited resources, i.e. limited storage, and the synchronization requirement makes it more difficult to administer.

6 Conclusion

In this work we demonstrated that PIR can be used to protect the user's interest. PIR provides strong protection – and potentially perfect protection if the modifications suggested in this work[5] are used. Thus, Web-PIR is an alternative to the MIX-based

[5] Not all simple details are given in this work, e.g. the requested pages should be have the same length. But this is also valid for MIXes and the general response to this problem is to add additional padding bits.

solutions if only the reading function is considered. The following features can be identified comparing the applications of the both techniques (see Table 1).

Method	Protection	Client	Server	Trust model
MIX	Complexity theoretic approach	Organization of *n* clients	No modification	*N* MIX stations and *n* clients
PIR	Perfect	Spontaneous communication	New design	*N* Server[6]

Table 1: Difference between Mix and PIR.

As shown in the beginning of this work, the MIX technique provides at most a complexity-theoretic protection. All known implementations are far behind this protection goal. Another drawback is the trust model: The more trust is required by the system, the more attack possibilities will arise. The MIX technique has to trust *n* other clients and N servers. Particularly, the organization of *n* other clients is a critical part here (for this function a global public key infrastructure is needed) if the maximum protection strength is envisaged. All of the deployments known to us neglect this important point.

The great advantage of the MIX method is that it can be used for both sending and receiving of messages. Since there is no need for a major modification of all servers, potentially all servers in the Internet could be addressed. The general applicability is a major drawback of PIR, since the server has to be redesigned. But for some servers providing some critical services (information), PIR can provide perfect protection of the user's privacy.

References

1. A. Ambainis: Upper Bound on the Communication Complexity of Private Information Retrieval, ICALP, LNCS 1256, Springer-Verlag, Berlin 1997.
2. A. Beimel, Y. Isahi, T. Malkin: Reducing the Servers Computation in Private Information Retrieval PIR with Preprocessing, CRYPTO 2000, LNCS 1880, Springer-Verlag, 2000.
3. A. Beimel, Y. Isahi, T. Malkin, and E. Kushilevitz: One-way functions are essential for single-server private information retrieval, In Proc. of the 31st Annu. ACM Symp. on the Theory of Computing (STOC), 1999.
4. O. Berthold, S. Clauß, S. Köpsell, A. Pfitzmann: Efficiency Improvements of the Private Message Service, 4th International Information Hiding Workshop, PA, USA 25. 4. 2001.
5. O. Berthold, H. Federrath, S. Köpsell: Web MIXes: A System for Anonymous and Unobservable Internet Access, IWDIAU, LNCS 2009, Springer-Verlag, 2001.
6. C. Cachin, S. Micali, M. Stadler: Computationally private information retrieval with polylogarithmic communication, In EUROCRYPT '99, LNCS 1592, Springer, 1999.

[6] Again, if perfect security is not envisaged then PIR does not need to trust the server [3, 6, 20].

7. D. Chaum: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms, Communications of the ACM 24/2 (1981).
8. D. Chaum: The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability, Journal of Cryptology 1/1 (1988).
9. B. Chor, O. Goldreich, E. Kushilevitz, M. Sudan: Private Information Retrieval, Proc. of the 36th Annual IEEE symposium Foundations of Computer Science, 1995.
10. B. Chor, N. Gilboa: Computationally Private Information Retrieval, 29th Symposium on Theory of Computing (STOC) 1997, ACM, New York 1997.
11. D. Cooper, K. Birman: The design and implementation of a private message service for mobile computers, Wireless Networks 1, 1995.
12. L. Cottrell: MIXmaster and Remailer Attacks, http://www.obscura.com /~loki/remailer/remailer-essay.html, 2001.
13. G. Di Crescenzo, Y. Ishai, R. Ostrovsky: Universal Service-Providers for Private Information Retrieval, Journal of Cryptology 14, 2001.
14. T. Demuth, A. Rieke: JANUS: Server-Anonymität im World Wide Web, Sicherheitsinfrastrukturen, Vieweg Verlag, 1999 (DuD-Fachbeiträge).
15. D. J. Farber, K. C. Larson: Network Security Via Dynamic Process Renaming, 4th Data Communications Symposium, 7-9 Oktober 1975, Quebec City, Canada.
16. C. Gülcü, G. Tsudik: Mixing Email with Babel, Proc. Symposium on Network and Distributed System Security, San Diego, IEEE Comput. Soc. Press, 1996.
17. P. A. Karger: Non-Discretionary Access Control for Decentralized Computing Systems, Master Thesis, MIT, , Mai 1977, Report MIT/LCS/TR-179.
18. D. Kesdogan: Privacy im Internet, Vieweg Verlag, ISBN: 3-528-05731-9, 1999.
19. J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, A. S. Tomkins: The Web as a graph: measurements, models, and methods, Proc. 5th Annual Int. Conf. Computing and Combinatorics, (1999).
20. E. Kushilevitz and R. Ostrovsky: Replication is not needed: Single database, computationally-private information retrieval, In IEEE FOCS '97, 1997.
21. R. Mathar, D. Pfeifer: Stochastik für Informatiker, Teubner, Stuttgart, 1990.
22. R. Ostrovsky, V. Shoup: Private Information Storage, STOC 1997, ACM, New York 1997.
23. A. Pfitzmann: Diensteintegrierende Kommunikationsnetze mit teilnehmer-überprüfbarem Datenschutz, IFB 234, Springer-Verlag, 1990.
24. A. Pfitzmann, M. Waidner: Netw. without user observability, Computers&Security 6/2, 87
25. M.G. Reed, P.F. Syverson, D.M. Goldschlag: Anonymous Connections and Onion Routing, Proc. of the 1997 IEEE Symposium on Security and Privacy, Mai 1997.
26. J. F. Raymond: Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems, IWDIAU, LNCS 2009, Springer-Verlag, 2001.
27. M. K. Reiter, A. D. Rubin: Crowds: Anonymity for Web Transactions, ACM Transactions on Information and System Security, Volume 1, 1998.
28. C. Rackoff, D. R. Simon: Cryptographic defense against traffic analysis, In 25th Annual ACM Symposium on the Theory of Computing, Mai 1993.
29. C. E. Shannon: Communication Theory of Secrecy Systems; The Bell System Technical Journal, Vol. 28, No. 4, Oktober 1949.
30. S. W. Smith, D. Safford: Practical Server Privacy with Secure Coprocessors, IBM Systems Journal. http://www.cs.dartmouth.edu/~sws/papers/
31. M. Waidner: Unconditional Sender and Recipient…, Eurocrypt '89, LNCS 434, 1990.
32. Zero-Knowledge-Systems, Inc.: http://www.freedom.net/ (2001).