A Lattice-Based Computationally-Efficient Private Information Retrieval Protocol

(Extended version of WEWORC paper, presented in July 2007, in Bochum, Germany)

Carlos Aguilar-Melchor and Philippe Gaborit

XLIM - Université de Limoges, France Carlos.Aguilar@xlim.fr,Philippe.Gaborit@xlim.fr

Abstract. A PIR scheme is a scheme that allows an user to get an element of a database without giving any information about what part of the database he is interested in.
In this paper we present a lattice-based PIR scheme, using an NTRU-like approach, in which the computational cost is a few thousand bit-operations per bit in the database. This improves the protocol computational performance by two orders of magnitude when compared to existing approaches. Our scheme has worse communication performance than other existing protocols, but we show that practical usability of PIR schemes is not as dependent on communication performance as the literature suggests, and that a trade-off between communication and computation leads to much more versatile schemes.

1 Introduction

The early publications on PIR [1, 2] present schemes that provide information theoretic privacy, and need a database replicated over different sites. Computationally Private Information Retrieval schemes were introduced by B. Chor and N. Gilboa [3] in 1997. They reduced the expansion factor on communications of PIR schemes, by considering computational privacy instead of information theoretic privacy. By computational privacy we mean that privacy is guaranteed against computationally bounded attackers. Using the same attacker model, Kushilevitz and Ostrovsky [4], presented the same year a scheme that provided computational privacy with an expansion factor similar to the one of Chor and Gilboa but without the need to use a replicated database. In this paper we will focus on this sort of protocols also known as single-database PIR protocols. Every PIR scheme that will be presented or cited in the following is a single-database PIR scheme.

Julien P. Stern presented in 1998 a generic construction [5] using group-homomorphic encryption schemes which led to the first protocols with acceptable communication performance.

One year after Stern's proposal, Cachin, Micali, and Stadler presented a scheme [6] based on a new trapdoor predicate that they called the ϕ-assumption. With this scheme, query size is almost independent of the number of elements in the database (growth is poly-logarithmic), and even if the system is not practically implementable[1], when database size increases this approach beats asymptotically all the PIR schemes published before it.

In 2004, there was a rediscovery of Stern's proposal [7], and a proposition by Lipmaa [8] which is basically Stern's construction with the recently discovered length-flexible homomorphic encryption scheme of Damgärd and Jurik [9]. In his paper Lipmaa twists Stern's construction, taking profit of the length-flexible cryptosystem to provide PIR schemes that are both practical and asymptotically interesting. Lipmaa remarks that using correctly this cryptosystem, it is possible to obtain a linear growth in the server reply (instead of exponential) when using recursively the PIR scheme. This greatly improves the versatility of the protocol and leads to an asymptotic behavior much better than with any of the previous schemes.

[1] Queries are too large and the communication rate is too small for almost any application.

Recently, the approach initially proposed by Cachin et al. led to a very interesting variation. In 2005, Gentry and Ramzan presented a scheme [10], which like Lipmaa's scheme is practical and presents an asymptotical improvement, even if for many applications Lipmaa's construction is better from a communication point of view. In their paper, the authors present a construction that generalizes the proposal of Cachin et al., and their scheme can be implemented using a slight variation of the ϕ-assumption.

The main performance measure used for these schemes is communication cost, disregarding computational cost. Thereby, current single-database PIR schemes provide almost optimal communication cost but require the database to use an enormous amount of computational power. This limits greatly the usability of these schemes, as even high-end servers are unable to generate PIR replies in a reasonable time for anything but the smallest databases.

In this work we present a lattice-based PIR scheme, using an NTRU-like approach, in which the computational cost is a few thousand bit-operations per bit in the database. This improves the protocol computational performance by two orders of magnitude when compared to existing approaches. Our scheme has communication performance not as good as other existing protocols, but eventually its far better computational performance permits to make our PIR protocol much more usable comparing to previously known protocols which are very difficult to use.

2 Basic concepts

We describe a database as a set of n l-bit elements. PIR requests are usually formed of a set of n query elements, one per each database element. Each of these query elements is combined with the database element it is associated to, and then the results are combined between them to obtain the PIR reply.

Because of this common approach, some techniques can be used with all the existing PIR schemes. In this section we present first how it is possible to adapt any scheme for any database element size, and second how the recursive usage of PIR schemes leads to much more versatile protocols. These techniques are not contributions of this paper, but it is important to be aware of their existence. In particular, in our protocol our query size seems to be linear in the number of elements of the database. The recursive usage of the protocol as shown in section 2.2 makes possible to reduce it to $O(n^{1/d})$ for a parameter d, just as it does for all the other existing PIR schemes.[2]

2.1 Iterative reply generation

It is straightforward to adapt a single-database PIR protocol to any value of l. For example, if a given single-database PIR scheme allows to recover one-bit elements from a database, it can also used to obtain 2-bit elements. When the user sends the PIR request to the database, this one will operate as follows :

- it generates a PIR reply by using the request over the set of n 1-bit elements formed by the first bit of each element in the database,
- it generates a second PIR reply by using the same request over the set of n 1-bit elements formed by the second bit of each element in the database.

[2] Except for the PIR schemes based on the ϕ-hiding assumption [6, 10].

Of course, this can be generalized to elements of any size and schemes allowing to retrieve chunks of information of arbitrary size. This is possible because the requests generated by the single-database PIR schemes are almost always independent from the database contents. When l is larger than the chunk size a scheme allows to retrieve, we will say that the database replies iteratively until the entire l-bit element is sent.

2.2 Load balancing and recursive usage

If the user sends n query elements and the database a PIR reply, the total communication cost is $O(n)$. To reduce this cost, it is possible to use a load balancing technique which was originally presented in the seminal paper about PIR [1]. The idea is to see the n-element database as a matrix of $\sqrt{n}$ lines and $\sqrt{n}$ columns each scalar in the matrix being a database element. The user sends $\sqrt{n}$ query elements, one for every column in the matrix, and the database replies iteratively sending back $\sqrt{n}$ PIR replies, one for each line in the matrix. As Figure 1 shows, with such an approach, the user retrieves a full column of data, containing the element he is interested in with total communication cost in $O(\sqrt{n})$.

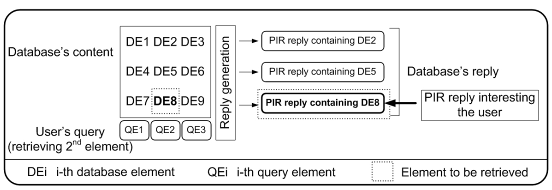

Figure 1. Load balancing.

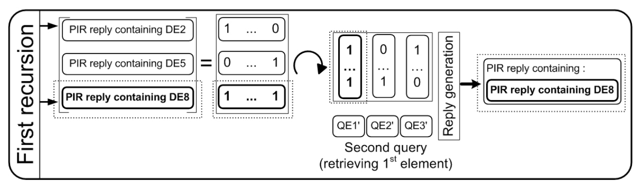

Figure 2. Recursive usage of a PIR scheme.

When representing the database as a matrix, instead of using the load balancing technique, it is possible to use the PIR scheme recursively as Kushilevitz and Ostrovsky proposed in [4]. The recursive usage of PIR schemes allows to lower the size of PIR requests while increasing the size of the PIR reply in a very versatile way. The main idea is that using the load balancing technique database obtains $\sqrt{n}$ PIR replies and each of them can be seen as an element of a *virtual* database. The user can therefore send a second query to retrieve one of the replies issued from the load balancing process.

In figure 2, we go back to the example given in figure 1 : the first recursion results in three PIR replies that are used as a virtual database for the second recursion. As with the load balancing technique, when recursion is used the query size shrinks (from $O(n)$ to $O(n^{1/2})$), and the reply size increases.

This approach is much more interesting than the load balancing technique for two reasons. The first one was not obvious at the time the scheme was proposed, and comes from the fact that some schemes can implement the recursion in a very efficient way as we will show later. The second reason is that load balancing may be done just once, representing the database as a matrix. Recursion can be done as many times as the number of dimensions that the database representation has.

If the database is represented as a cube of size $n^{1/3}$, the user will send three queries with $n^{1/3}$ elements each. The database will compute a matrix of $n^{1/3} \times n^{1/3}$ PIR replies from the first query. This matrix can be seen as a virtual database and the second query will be used to retrieve one column of $n^{1/3}$ PIR replies. This column will also be used as a virtual database of $n^{1/3}$ elements, and the third query will be used to obtain the PIR reply containing the element the user is interested in. Generally, if the database is represented by a d-dimension hyper-cube, d recursions are possible. With such a representation the user will send d requests, each composed of only $n^{1/d}$ residues. This allows a user to shrink greatly the queries' size, but the size of the PIR replies increases quickly (exponentially in most cases) in d. A trade-off must be made, depending on the application the PIR scheme is used for.

Some PIR schemes [10, 11] do a long pre-computation over the database contents before answering to the PIR queries. Sometimes this pre-computation is mandatory for the scheme to work properly [10], other times it just brings a performance improvement [11]. When using recursion, pre-computation cannot be done over the virtual intermediate databases, as they depend on the users' first queries, and therefore these two techniques are incompatible. Whenever a scheme has a phase of pre-computation, we will thus not consider the usage of recursion.

3 Description

3.1 High-level overview of our protocol

The PIR scheme we propose relies on the simple idea of controlled noise addition. The main idea is to to start from a secret random $[N, 2N]$ matrix M of rank N over a field Z/pZ. This matrix is used to generate a set of different matrices obtained by multiplication on the left side by invertible random matrices. These matrices (which can also be seen as lattices by joining pI_{2N} for I_{2N} the identity $2N \times 2N$ matrix) are disturbed by the user by the introduction of noise in half of the matrices' columns to obtain respectively softly disturbed matrices (SDMs) and hardly disturbed matrices (HDMs).

To obtain an element from the database the user sends a set of SDMs and one HDM. The database inserts each of its elements in the corresponding matrix with a multiplicative operation OP and sums all the rows of the resulting matrices to obtain the database reply, a single noisy vector. Using the unmodified columns of the matrices sent in the request, the user is able to find the noise associated to the returned noisy vector. If the soft noise multiplied by the total noise factor (which is proportional to the number of elements in the database) is much smaller than the hard noise, it can be filtered out and the user can retrieve the information associated to the noise of the HDM matrix. The scheme uses the same kind of idea that for the lattice-based NTRU cryptosystem: one considers a vector space over a field Z/pZ where the key idea is to control an error by keeping it non altered by any modular operation.

3.2 Request generation

The scheme will have three global integer parameters: $2N$, the dimension of the lattice and special parameters p and q. The database is described as a set of n l-bit elements, and we note i_0 the index of the database element the user is interested in. To obtain a PIR request, the user will follow:

Protocol 1

---

1. Note $l_0 = \lceil log(n \times N) \rceil + 1$ and set q as 2^{2l_0} and p as a prime larger than 2^{3l_0}.
2. Generate A and B, two random matrices over Z/pZ such that A is invertible, and note $M = [A|B]$.
3. For each $i \in \{1 \cdots N\}$ compute a matrix $M_i^{''} = [A_i|B_i]$ by multiplying M by a random invertible matrix P_i.

4. Generate the random scrambling matrix Δ as a $N \times N$ random diagonal matrix over Z/pZ.
5. For each $i \in \{1 \cdots N\} \backslash i_0$ generate the soft noise matrix D_i, a $N \times N$ random matrix over $\{-1, 1\}$, and compute the soft disturbed matrix $M_i^{'} = [A_i | B_i + D_i \Delta]$.
6. Generate D_{i_0}, the hard noise matrix, by:
   – generating a soft noise matrix,
   – replacing each diagonal term by q.
7. Compute the hard disturbed matrix $M_{i_0}^{'} = [A_{i_0} | B_{i_0} + D_{i_0} \Delta]$.
8. Choose a random permutation of columns $\mathcal{P}(\cdot)$ and compute $M_i = \mathcal{P}(M_i^{'})$ for $i \in \{1 \cdots n\}$.
9. Send the ordered set $\{M_1 \cdots M_n\}$ to the database.

3.3 Answer encoding

To answer to the PIR reply the database follows protocol 2. The result is a vector V of dimension $2N$ over Z/pZ. We will suppose that each element has exactly the number of bits that can be encoded into a matrix ($l_0 \times N$ bits).

Protocol 2

---

1. Split each database element m_i in N l_0-bit integers $\{m_{i1} \cdots m_{iN}\}$.
2. For each $i \in \{1 \cdots n\}$ construct the vector $v_i = \sum_{j=1}^{N} m_{ij} M_{ij}$ where M_{ij} denotes the j-th row of M_i.
3. Return $V = \sum_{j=1}^{n} v_i$

3.4 Information extraction

To extract the information from the database reply, the user will operate in two phases. First he will recover the noise included in the vector (steps 1 and 2 of protocol 3, and then he will unscramble and filter out this noise to obtain the information (steps 3 to 5).

Protocol 3

---

1. Compute the non-permuted noisy vector $V^{'} = \mathcal{P}^{-1}(V)$.
2. Retrieve $E = V_D^{'} - V_U^{'} A^{-1} B$, the scrambled noise, $V_U^{'}$ and $V_D^{'}$ being resp. the undisturbed and disturbed halves of $V^{'}$.
3. Compute the unscrambled noise $E^{'} = E \Delta^{-1}$
4. For each $e_j^{\prime}$ in $E^{'} = [e_1^{\prime} \cdots e_n^{\prime}]$, compute $e_j^{\prime\prime} = e_j^{\prime} - \epsilon$ with $\epsilon := e_j^{\prime} \% q$ if $e_j^{\prime} \% q < q/2$ and $\epsilon := e_j^{\prime} \% q - q$ else.
5. For each $j \in \{1 \cdots n\}$, compute $m_{i_0 j} = e_j^{''} q^{-1}$.

To recover the noise, the user will first undo the random column permutation (step 1). Then he will use the N first coordinates of the vector and the initial matrix M to obtain what the N last coordinates (which have been disturbed) should be without noise. He will use these values to extract the noise inserted in these coordinates (step 2).

This noise is composed of soft and hard noise, but it cannot be directly filtered because it was scaled up by the noise scrambling matrix. The user will therefore first eliminate this scrambling (step 3). He will then filter out the soft noise (step 4), and divide each coordinate by the hard noise factor to obtain the database sub-elements of m_{i_0}.

Extraction correctness :
Noting ϵ_{ijk} the (j,k)-th coordinate of D_i, the unscrambled error vector coordinates can be expressed as:

$$e_j^{'} = \sum_{i \in \{1 \cdots n\} \backslash i_0} \sum_{k=1}^{N} m_{ik} \epsilon_{ijk} + m_{i_0 j} q,$$

we will therefore have $e_j^{'} = A + m_{i_0 j} q$ with $|A| = |\sum_{i \in \{1 \cdots n\} \backslash i_0} \sum_{k=1}^{N} m_{ik} \epsilon_{ijk}| < 2^{l_0} \times n \times N$ and as $n \times N \leq 2^{l_0 - 1}$ we obtain $|A| < q/2 = 2^{2l_0 - 1}$. It results that after step 4 the user obtains $e_j^{''} = m_{i_0 j} q$ for each $j \in \{1 \cdots n\}$. As $m_{i_0 j} \times q < p = 2^{3l_0}$ the user retrieves all of the sub-elements of m_{i_0} at the end of the extraction.

□

3.5 A toy example

We give here a toy example of our protocol. Let $n = 2$, $N = 2$, $l = 6$, and $i_0 = 2$. Figure 3 gathers the operations done for query generation.

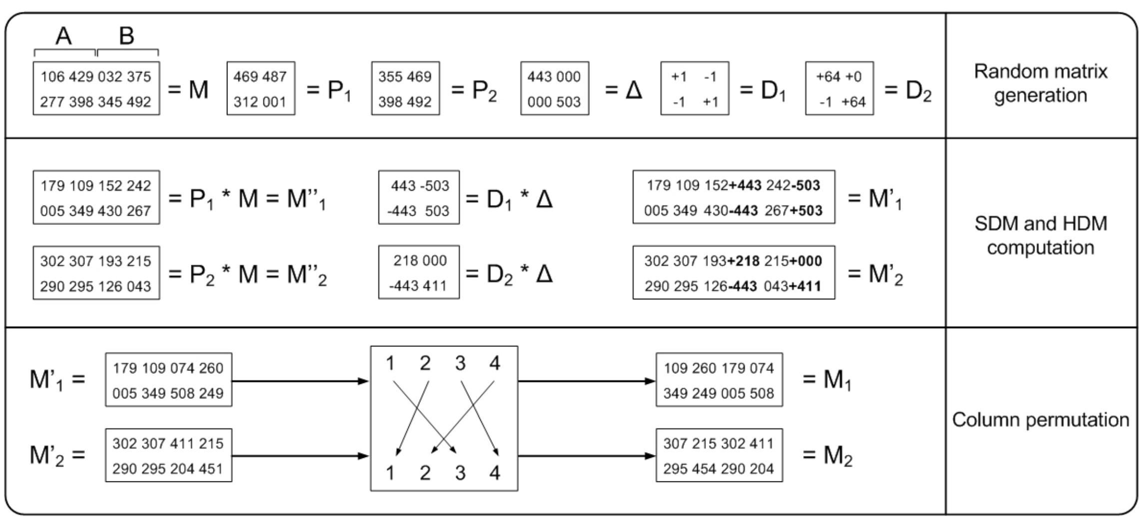

Figure 3. Query generation.

The user first sets $l_0 = \lceil \log(2 \times 2) \rceil + 1 = 3$, $q = 2^{2l_0} = 64$, and $p = 521 > 2^{3l_0}$. He then generates A and B, two random 2×2 matrices over $GF(521)$, A being invertible. After that he generates two invertible matrices P_1 and P_2, the noise scrambling diagonal matrix Δ, one soft noise matrix D_1 and one hard noise matrix D_2.

Then the user computes $M_1^{''}$ and $M_2^{''}$ multiplying M byP_1 and P_2. The first of these matrices is softly disturbed by adding the results of the scrambled soft noise matrix $D_1 \Delta$ to its third and fourth columns and results in $M_1^{'}$. Similarly, $M_2^{'}$ is obtained from the addition of the scrambled hard noise matrix $D_2 \Delta$ to $M_2^{''}$. Finally the user chooses a random permutation of the four columns and applies it both to $M_1^{'}$ and $M_2^{'}$ to obtain the final query $\{M_1, M_2\}$.

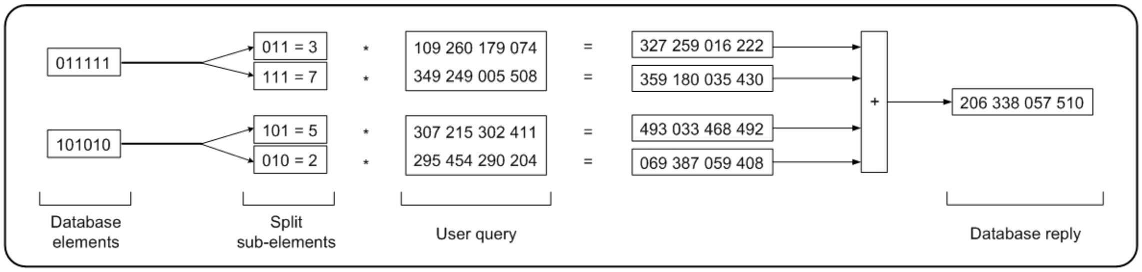

Figure 4. Reply generation.

The database has two 6-bit elements, each is split in two 3-bit sub-elements. This sub-elements will be represented as integers in the interval $\{0 \cdots 7\}$ in figure 4. When the database receives the query $\{M_1, M_2\}$, it multiplies these matrices' rows by the sub-elements and sends back the sum of all results.

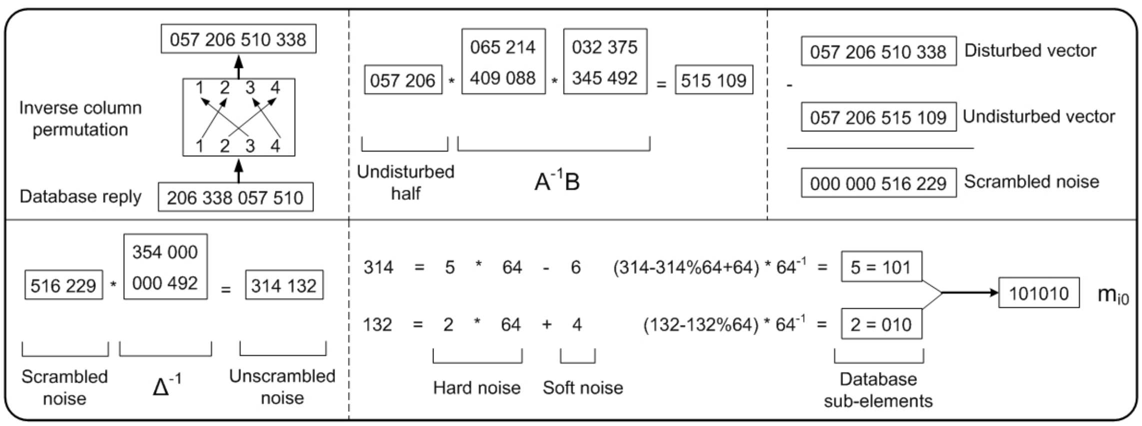

Figure 5. Information extraction.

Upon reception of the database reply, the user inverts the column permutation (see figure 5). He then deducts from the first columns the undisturbed vector associated to the database reply and uses it to recover the scrambled noise. Using Δ^{-1} he unscrambles the noise and through the Euclidean algorithm he recovers the databases sub-elements encoded in the answer and reconstructs m_{i_0}.

4 Security

In our scheme, an attacker able to distinguish between HDMs and SDMs will also be able to distinguish the different sort of queries, and therefore the users' privacy will not be preserved. In this section, we first discuss about the structural security of our scheme, considering whether or not an attacker is able to break completely the system by retrieving the private data such as the permutation matrix. To deal with this security we introduce in section 4.1 a new problem, *the Hidden Lattice Problem (HLP)*. We discuss its relationship with a well known NP-complete problem, the *Punctured Code Problem*, and evaluate its practical security. Finally, we show that our scheme's structural security is equivalent to HLP.

The second security issue we discuss is the distinguishability attack, common to all PIR schemes. Of course, if one is able to break the Hidden Lattice Problem, and therefore the structural security of the system, one is also able to distinguish between the HDMs and SDMs forming the queries. On the other hand, distinguishing HDMs from SDMs may be easier than breaking the structural security. Therefore, we introduce in section 4.2 a problem related to the distinguishability of HDMs and SDMs, the *Differential Hidden Lattice Problem (DHLP)*. We show that it is likely to be as hard as HLP and in particular that lattice based attacks like LLL seem inefficient in that case.

Eventually, in section 4.4 we propose a set of parameters which provide good security and practical usability.

4.1 Structural security: The Hidden Lattice Problem

Definitions and theoretical security In this section we define a new lattice-based problem on which the structural security of our scheme relies. We justify the fact that it is very likely a hard problem by relating it to another, well-known, NP-complete problem.

Definition 1 Hidden Lattice Problem
Let V be a k dimensional vector space of length n over a finite field $GF(p)$ for p a large prime number. Consider a set of r different random basis $\{V_1 \cdots V_r\}$ of V with $V_i = [V_{i,1}| \cdots |V_{i,n}]$. Fix randomly a subset of s columns such that its complementary set $S = \{j_1 \cdots j_{n-s}\}$ holds k independent columns. Choose randomly $i_0 \in \{1, \cdots, r\}$ and $q \in GF(p)$ with $1 \ll q \ll p$. For each V_i generate a set of random columns $\{R_{i,1} \cdots R_{i,n-k}\}$ such that $R_{i,j}$ is composed of elements in $\{-r_j, r_j\}$ (r_j being a random element of $GF(p)$). For each $l \in \{1, \cdots, r\}$ multiply the l-th coordinate of $R_{i_0,l}$ by q. Disturb each V_i into V_i' by adding these random columns to $\{V_{i,j_1} \cdots V_{i,j_{n-k}}\}$.
Deduce from the set of disturbed basis which are the $n-k$ disturbed columns.

A query from our PIR scheme is an instance of HLP with $k = s = N$, $n = 2N$, $q = 2^{l_0}$ and $p > 2^{3l_0}$ with $l_0 = \lceil \log(n_{DB} \times N) \rceil + 1$ and n_{DB} the number of elements in the queried database. The associated assumption for our scheme's structural security (that may be noted the hidden lattice assumption or HLA) is that there is no family of circuits with polynomially bounded size in N and $\log(p)$ able to solve these instances of HLP with non-negligible advantage.

Definition 2 Punctured Code Problem *(proved NP-complete by Wieschebrink in [12])*
Let M be a $k \times n$ matrix, H a $k \times m$ matrix [of rank k][3] with $k \leq m \leq n$, both over a field K. Does there exist a non-singular matrix T and a subset S of $\{1..n\}$ with $|S| = n - m$ such that the code TM_S obtained by the deletion of the columns of S equals the code H ?

A circuit $\mathcal{A}$ able to find the subset S when possible and returning $\emptyset$ when this subset doesn't exist solves this problem. Reciprocally, if a circuit $\mathcal{A}'$ solves the Punctured Code Problem, it is easy to simulate a circuit that finds the subset S when possible and returns $\emptyset$ when this subset doesn't exist. Indeed, deleting one by one the columns of M and querying $\mathcal{A}'$ leads to finding S by a simple test and trial method. This problem is therefore equivalent to what we will call the Code Puncture Search Problem.

[3] In the problem presented by Wieschebrink the matrices are not necessarily supposed to be of rank k, but the proof given in his paper remains correct with this constraint.

Definition 3 Code Puncture Search Problem
Let H be a $k \times m$ matrix of rank k and M a disturbed $k \times (m+s)$ matrix obtained by multiplying H by a random non-singular $k \times k$ matrix T and by adding to it s random columns in between the m columns of H.

Deduce from these two matrices which are the s random columns of M.

Equivalence between HLP and our scheme's structural security The structural security of our scheme relies on the secrecy of the hidden lattice described by M, the scrambling matrix Δ and the permutation $\mathcal{P}(\cdot)$. Thus, if an attacker is able to break completely our system he will obtain the secret permutation and therefore solve the corresponding instance of HLP by retrieving the indexes of the disturbed columns. Conversely, when the indexes of the disturbed columns are known it is possible to recover a basis of the hidden lattice and the scrambling matrix Δ.

Suppose an adversary knows the N non modified columns, without lack of generality we can suppose they are the first N columns. Suppose we have 3 disturbed matrices M_1, M_2 and M_3. We have $M_1 = P_1 M + [0|D_1\Delta]$ and $M_2 = P_2 M + [0|D_2\Delta]$ for P_1 and P_2 the random matrices used to construct M_1 and M_2.

Lemma 1. *If an adversary knows the non modified columns he can recover $P_2 P_1^{-1}$.*

Proof. Write $M = [A|B]$. Since we know that the first N columns are not modified, just comparing the first N columns of M_1 and M_2 we recover $P_1 A$ and $P_2 A$ an the result follows from the evaluation of $P_2 A (P_1 A)^{-1}$.

□

Once $P_2 P_1^{-1}$ is known, the adversary computes $S_{12} = P_2 P_1^{-1} M_1 - M_2 = [0|P_2 P_1^{-1} D_1 \Delta] - [0|D_2\Delta]$. Using the last column of S_{12} and setting the last column of $D_1\Delta$ and $D_2\Delta$ as unknowns he gets a set of $2N$ unknowns and N equations. Repeating the same process with M_2 and M_3 brings N new unknowns and equations. Finally, repeating this process again with M_1 and M_3 brings a set of N new equations and no unknowns, leading the adversary to $3N$ equations and $3N$ unknowns, which can be solved in $O(N^3)$ operations. Repeating the same operation for the other columns, the adversary obtains all the elements of $D_1\Delta$. As the elements of D_1 are in $\{-1, 1\}$ retrieving Δ is straightforward and a basis of the hidden lattice is given by $M_1 - D_1\Delta$.

Practical considerations Notice that for the PIR scheme presented, the initial matrix is a non-singular $N \times N$ matrix to which N random columns are added. This is a very special situation and one may try to take advantage of this. The probability that a random $N \times N$ matrix over $GF(p)$ is singular is close to $\frac{1}{p}$, hence taking p with more than 60 bits makes this possibility overwhelmingly unlikely, and in any case testing matrix singularity until finding one such matrix would cost at least 2^{80} for $N = 50$. We can therefore suppose that for p and N large enough it is computationally infeasible to find one. Thus, in the Code Puncture Search Problem if p is large enough, it becomes impossible to find the set of s columns inserted as for any set S there exists a non-singular matrix linking M_S (and therefore all the corresponding $\{M'_{1S} \cdots M'_{rS}\}$)) to H. This will also be the case with the Hidden Lattice Problem and the adversary will only be able to test if a given subset is correct by analyzing the random modifications until finding a set leading to random.

Overall, if an adversary wants to find the undisturbed columns, it has to characterize in some way the use of $\{-\alpha, +\alpha\}$-type noise. This can be done in two ways: through the search for vectors of small norms, but we will see in section 4.3 that it does not seem possible, or analyzing the inserted randomness for every subset S. The number of subsets of N columns among $2N$ induces a search complexity of at least $\binom{2N}{N} \simeq 2^{2N}$ possibilities and therefore is computationally unfeasible for $N = 50$. Hence for this problem the best attack seems to be exponential. We will see in the next why lattice based attacks such as LLL seem to be inefficient.

4.2 Distinguishability: The Differential Hidden Lattice Problem

Consider now the following problem related to the distinguishability of queries in our PIR scheme.

Definition 4 Differential Hidden Lattice Problem
Let V be a k dimensional vector space of length n over a finite field $GF(p)$ for p a large prime number and T_1, T_2 two different subsets of $\{1, \cdots, r\}$ with t_1 and t_2 elements. Consider a set of r different random basis $\{V_1 \cdots V_r\}$ of V with $V_i = [V_{i,1}| \cdots |V_{i,n}]$. Fix randomly a subset of s columns such that its complementary set $S = \{j_1 \cdots j_{n-s}\}$ holds k independent columns. Choose randomly $q \in GF(p)$ with $1 \ll q \ll p$, $r \in \{1, 2\}$ and set $T = T_r$. For each V_i generate a set of random columns $\{R_{i,1} \cdots R_{i,n-k}\}$ such that $R_{i,j}$ is composed of elements in $\{-r_j, r_j\}$ (r_j being a random element of $GF(p)$). For each $l \in \{1, \cdots, r\}$ and each $i \in T$ multiply the l-th coordinate of $R_{i,l}$ by q. Disturb each V_i into V_i' by adding these random columns to $\{V_{i,j_1} \cdots V_{i,j_{n-k}}\}$.
Deduce from T_1, T_2 and the set of disturbed basis the value of r.

Let $\{V_1', \cdots, V_{n_{DB}}'\}$ be a query of our PIR scheme for an element of index i_0 and T_1, T_2 two different subsets of $\{1, \cdots, n_{DB}\}$ such that $t_1 = t_2 = 1$ and $i_0 \in T_1 \cup T_2$. This query, attached with T_1, T_2, is an instance of DHLP with parameters $k = s = N$, $n = 2N$, $q = 2^{l_0}$ and $p > 2^{3l_0}$ with $l_0 = \lceil \log(n_{DB} \times N) \rceil + 1$ and n_{DB} the number of elements in the queried database.

The assumption associated to user privacy in our scheme (that may be noted the differential hidden lattice assumption or DHLA) is that there exists no family of circuits with polynomially bounded size in N and $\log(p)$ able to solve these instances of DHLP with non-negligible advantage for any two subsets such that $t_1 = t_2 = 1$. Of course DHLP is an easier problem than HLP:

Proposition 1 *If an adversary is able to solve HLP with non-negligible advantage he is also is able to solve DHLP for any t_1, and t_2 with non-negligible advantage .*

This result is straightforward since solving the HLP problem permits to recover the different perturbations and point out the multiplication by q in the basis corresponding to an index in T_1 or T_2.

4.3 Distinguishability: Lattice based attacks

To break user privacy, an attacker just needs to distinguish SDMs from HDMs.

The matrices V_i' forming the query can be obviously seen as $[2N, N]$ codes over $GF(p)$. Meanwhile, since the basic mechanism of our system is to be able to make a difference between an addition of noise of type $\{-1, 1\}$ and a greater noise of type $\{-q, q\}$, the context of coding theory is not adapted. Indeed, for linear codes the main tool is the Hamming weight, which makes a difference only between 0 and elements of $GF(p)^*$. In our case, we are interested by a more precise weight, which would make a difference between all the elements of $GF(p)^*$. The adapted context in our case is hence to consider lattice theory and the Euclidean distance, which permits to reach this distinguishability between elements.

The explicit link between a matrix over $GF(p)$ and a lattice is made through the well known Construction A by row concatenating to any matrix over $GF(p)$ a matrix identity times p. From each matrix V_i' forming the query, we obtain an associated lattice L_i, and a distinguishability attack would consist in pointing out the hard noise lattice L_{i_0} (V_{i_0}') among the other lattices L_i (V_i').

The main tool for lattice based attacks is the famous LLL algorithm. The general idea is to characterize a target vector (typically a solution to a problem) as a short vector of a lattice. The attack is done in two steps:

1. Build a lattice such that a short vector of this lattice is a solution to the problem considered.
2. Run the LLL algorithm to recover the short vector and hence the solution.

In our case, we want to make a difference between HDMs and SDMs applying the same approach. For instance, by showing that the perturbation of SDMs is made only by elements in $\{-1, 1\}$ whereas it contains also elements in $\{-q, q\}$ for HDMs. We believe that in our case, LLL based attacks are not applicable since it does not seem possible to characterize a solution of our problem as a shortest vector. It may be possible to prove that a solution vector belongs to some lattice, but with a norm higher than the expected norm by the Gaussian heuristic, which makes this characterization and hence the LLL attack *a priori* not possible.

In the following we first give two arguments to explain why the norms of our solution vectors seem difficult to characterize as shortest vectors of associated lattices. Secondly, we examine methods used for classical lattice cryptanalysis, to show that they do not seem to be applicable in our case.

An easy obvious false attack At first glance, our problem may seem easy to break by the following attack, which is not valid in practice since it does not take into account some of the features of our system.

Suppose indeed that we are given a first $N \times 2N$ matrix $M_1 = [A|B]$ and then a second matrix $M_2 = [HA + R_1|HB + R_2]$, such that H is an invertible matrix and the matrix $[R_1|R_2]$ is a matrix with only N columns of elements in $\{-1, 1\}$. This problem corresponds to the case when the hidden lattice is known (but not the position of the error columns), and when no scrambling matrix Δ is used. In that case, it is possible to recover the position of the error columns by taking the first row of M_2, $x = (a' + r_1|b' + r_2)$ (with $(r_1|r_2)$ the first row of $[R_1|R_2]$), and building a lattice L formed by row concatenation of the matrix M_1, the first row of M_2 and p times an identity matrix. Indeed, it is then obvious to see that the vector $r = (r_1|r_2)$ belongs to the lattice L. Using the last column of H, h, we build the vector $y = (h^T| - 1|0...0)$ which multiplied by the lattice results in r. The vector r allows to characterize our solution (it gives the disturbed columns) and, since r is composed only of elements in $\{-1, 1\}$, it is possible to show that this is a shortest vector of L and thus apply LLL to recover it.

Variations on this toy attack: the system in practice In practice, our scheme cannot be attacked like this for two reasons.

First, when multiplying the inserted noise by a scrambling matrix Δ (a diagonal matrix with non null elements), it is possible to do the same attack but this time the resulting vector r is not composed anymore with elements in $\{-1, 1\}$ but with random elements of $GF(p)^*$. This makes r impossible to characterize as a shortest vector since it has, on the average and with strong probability (exponentially close to 1), no reason to be a shortest vector. Indeed, the non null elements δ_i of Δ are random, and having zeros on half of the coordinates and random elements on the second half is not enough to be a shortest vector.

Second, for this previous attack, we considered two matrices M_1 and M_2 such that M_1 corresponded to the hidden lattice (the lattice with no perturbation), and M_2 corresponded to a disturbed matrix. In practice, one has *two* disturbed matrices $M_1 = [HA + R_1|HB + R_2]$ and $M_2 = [H'A + R'_1|H'B + R'_2]$ (we do not consider the scrambling matrix Δ for this example) for H and H' two random invertible matrices and, $[R_1|R_2]$ and $[R'_1|R'_2]$ defined as previously. Let us adopt the same approach than for our obvious previous attack. Consider a lattice L built from M_1, the first row of M_2, $x = (a' + r'_1|b' + r'_2)$ and p times an identity matrix. As for the previous case, we want to recover a vector of type $(r'_1|r'_2)$. To be able to vanish $(a'|b')$ in x as in the obvious attack, this time rather than multiplying by the last column of H, we multiply by the last column h of $H'H^{-1}$. The issue in this case, is that doing this we also add a term coming from the $[R_1|R_2]$, and get a vector $y = (r'_1 + h^T.R_1|r'_2 + h^T.R_2)$. This vector is of the same type as before but pertubated by h (coming from the product of random matrices H' and H^{-1}), R_1 and R_2. Eventually, even if the vector y may have some coordinates set to zero, the remaining coordinates are random elements of $GF(p)^*$ because of the action of h (which is a random vector), and hence y cannot be characterized as a shortest vector.

Comparison with previously known lattice based attacks We have shown why lattice based attacks are very unlikely in our case. Although for lack of space we do not go into details, an examination of the lattices used for attack by LLL different lattice based cryptosystem, like Knapsack, GGH or NTRU (see [13] for a very nice survey) show that all previoulsy known approaches do not work for our system. In particular a Knapsack like type attack with a lattice of as the one in Figure 3, cannot work since there is no linear relation between the SDMs and HDMs, when it is the case for the target solution vector searched for the Knapsack lattice (see [13] for details).

$$L_K = \begin{bmatrix} 1 & 0 & \cdots & 0 & \frac{-a_1}{a_n} \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \cdots & 0 & 1 & \frac{-a_{n-1}}{a_n} \\ y_1 & y_2 & \cdots & y_{n-1} & y_n \end{bmatrix} \qquad L_{PIR} = \begin{bmatrix} 1 & 0 & \cdots & 0 & SDM_1 \\ 0 & \ddots & \ddots & \vdots & \\ \vdots & \ddots & \ddots & 0 & \cdots \\ 0 & \cdots & 0 & 1 & SDM_r \\ p & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & p \end{bmatrix}$$

Figure 6. The knapsack lattice and an equivalent construction for our scheme

4.4 Proposed parameters

For practical use we propose as parameters, $l_0 = 20$ $(3l_0 = 60)$, and $N = 50$. The parameter $N = 50$ implies a complexity of more than 2^{100} operations to retrieve the 50 non modified columns. Taking $l_0 = 20$ permits to make out of reach the search for non invertible submatrices of M_i (which is not in itself an attack but a first step in the direction of a potential attack). We propose to choose $p = 2^{60} + 325$. Notice that such a choice (a power of 2 plus a small integer) may also be used to fasten the computation of modular multiplication as for elliptic curves. A weak estimation of the number of elements which can be handled by such parameters is $n \leq n_{max} = \frac{2^{l_0}}{50} \simeq 20000$. Increasing l_0 linearly results in an exponential growth of n_{max} (one billion elements for $l_0 = 36$) and therefore for any reasonable size of n this parameter does not need major changes.

5 Performance comparison

Computational performance comparison is very simple. For Lipmaa's scheme, if $k = 1024$, the computational cost is roughly of one 2048-bit modular multiplication per bit in the database. For Gentry and Ramzan's scheme, the cost is a little bit lower, a 1365-bit modular multiplication per bit. For the given parameters, in our scheme the cost is one hundred additions of sixty bits per bit in the database.

Our dual Opteron 248 server can compute 2×10^5 2048-bit modular multiplications, and 4×10^5 1365-bit modular multiplications per second. By considering a 64 bits processor at 4 GHz, which can perform $4 \times 10^9 \times 64$ operations per second our scheme can deal with roughly 4×10^7 bits per second which is two orders of magnitude larger than Gentry and Ramzan's scheme.

As we already said in the introduction, in order to answer a query in a PIR scheme, the database must process all of its entries. Therefore, in an n element database, n bits must be processed to deal with one single bit of the element the user is interested in. The throughput a database will be able to generate will therefore be $2 \times 10^5/n$ bits/s with Lipmaa's scheme $4 \times 10^5/n$ bits/s with Gentry and Ramzan's scheme and $4 \times 10^7/n$ with our scheme. Even for databases with a small number of elements this results on small throughputs. For example, for $n = 1000$ we obtain respectively 200bits/s, 400bits/s and 40Kbits/s. Given today's available bandwidths, having a small expansion factor or not over this throughput values is secondary.

Scheme	d=1					d=2				
	Query		Download	Bandwidth Usage		Query		Download	Bandwidth Usage	
	size	time	time	exp. factor	percentage	time	size	time	exp. factor	percentage
Lipmaa	2Mb	2s	33h	2	0.002%	162Kb	0,16s	33h	3	0.003%
Gentry and Ramzan	3Kb	$\simeq$ 0s	17h	4	0.016%	3Kb	$\simeq$ 0s	17h	4	0.016%
Ours	300Mb	5min	10min	6	1.2%	19Mb	19s	10min	36	7.2%

Figure 7. Query and download times.

Figure 7 presents the query and download times as well as the bandwidth usage for retrieving a three Mbytes file (for example an mp3 song) from a one thousand elements database. The user is supposed to have a 20 Mbits/s download and 1 Mbit/s upload digital subscriber line to the Internet. Without recursion (parameter $d = 1$), sending the query takes five minutes with our scheme and is almost immediate with the previous protocols. However, the downloading phase is just ten minutes long with our scheme while it takes many hours with the other ones. Thus, queries are larger with our protocol and take more time to be sent, but the download phase is so much reduced that this issue is negligible. If recursion is used ($d = 2$), sending the query takes only 19 seconds but the bandwidth usage is increased from 1.2% to 7.2%.

Note that even if the download expansion factors for our scheme are large, bandwidth usage is reasonable. This usage is larger than with the previous schemes but remains small when compared with today's available bandwidths. Note also, that even if the download expansion factor of Lipmaa's scheme (resp. Gentry and Ramzan's scheme) was 1000 (resp. 200) the bandwidth usage for these schemes will be of 1%. Indeed, these protocols use very little bandwidth because they are so (computationally) costly that even a high-end server cannot generate a significant bandwidth. Having small expansion factors with so low throughputs is almost useless in practice. Number theory based schemes are close to an optimal communication cost with an expansion factor close to 1, but their relative slowness for the reply generation makes them almost unpractical when the lattice-based approach presents a real potential for many applications.

References

1. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private Information Retrieval. In: IEEE Symposium on Foundations of Computer Science (FOCS). (1995) 41–50
2. Ambainis, A.: Upper Bound on Communication Complexity of Private Information Retrieval. In: ICALP. (1997) 401–407
3. Chor, B., Gilboa, N.: Computationally Private Information Retrieval (Extended Abstract). In: STOC. (1997) 304–313
4. Kushilevitz, E., Ostrovsky, R.: Replication Is Not Needed: Single Database, Computationally-Private Information Retrieval (extended abstract). In: FOCS: IEEE Symposium on Foundations of Computer Science (FOCS). (1997) 364–373
5. Stern, J.P.: A New Efficient All-Or-Nothing Disclosure of Secrets Protocol. In Ohta, K., Pei, D., eds.: ASIACRYPT. Volume 1514 of Lecture Notes in Computer Science., Springer (1998) 357–371
6. Cachin, C., Micali, S., Stadler, M.: Computationally Private Information Retrieval with Polylogarithmic Communication. In: EUROCRYPT: Advances in Cryptology: Proceedings of EUROCRYPT. (1999) 402–414
7. Chang, Y.C.: Single Database Private Information Retrieval with Logarithmic Communication. In: ACISP: Information Security and Privacy: Australasian Conference. (2004) 50–61
8. Lipmaa, H.: An Oblivious Transfer Protocol with Log-Squared Communication. In Zhou, J., Lopez, J., Deng, R.H., Bao, F., eds.: ISC. Volume 3650 of Lecture Notes in Computer Science., Springer (2005) 314–328
9. Damgård, I., Jurik, M.: A Length-Flexible Threshold Cryptosystem with Applications. In: ACISP 2003. (2003) 350–364
10. Gentry, C., Ramzan, Z.: Single-Database Private Information Retrieval with Constant Communication Rate. In: ICALP: Annual International Colloquium on Automata, Languages and Programming. (2005) 803–815
11. Asonov, D., Freytag, J.C.: Almost optimal private information retrieval. In Dingledine, R., Syverson, P.F., eds.: Privacy Enhancing Technologies. Volume 2482 of Lecture Notes in Computer Science., Springer (2002) 209–223
12. Wieschebrink, C.: Two NP-complete Problems in Coding Theory with an Application in Code Based Cryptography. In: 2006 IEEE International Symposium on Information Theory. (2006) 1733–1737
13. Phong Q. Nguyen and Jacques Stern: The two faces of lattices in cryptology. In: Cryptography and Lattices, International Conference, CaLC 2001, Providence, RI, USA, March 29-30, 2001, Revised Papers. Volume 2146 of Lecture Notes in Computer Science., Springer (2001) 146–180
14. Lipmaa, H.: An Oblivious Transfer Protocol with Log-Squared Communication. In: The 8th Information Security Conference (ISC'05). Volume 3650 of Lecture Notes in Computer Science., Springer-Verlag (2005) 314–328