The Security of Aguilar-Melchor and Gaborit's Lattice-Based Private Information Retrieval Protocol

Richard Roberts, Dennis Sell, Terry Sun
{ricro,selld,terrysun}@seas.upenn.edu

March 30, 2015

Abstract

The lattice-based Private Information Retrieval (PIR) protocol by Aguilar-Melchor and Gaborit [4] provides an alternative to the number-theoretic schemes protocols for single-server computational PIR that came before it. With slightly greater communication costs, their scheme improves the computational performance by two orders of magnitude. However, this scheme has not been reduced to an NP-Hard problem, nor have lattice attack techniques been successful in attacking this protocol. Thus, little is known about the security of this protocol. In this paper, we first provide a summary of the scheme, its background, and its security analysis presented by the original authors. We then build upon their work in an attempt to determine security properties of the scheme.

1 Introduction

Private Information Retrieval (PIR) schemes are methods of accessing information from a database while at the same time hiding from the server which element is being accessed. There exist both single-server and multi-server PIR schemes, which fall generally into two camps: computational PIR, based on the assumption that a polynomially-bounded adversarial database cannot decipher the query that it's presented with, and information-theoretic PIR, which posits that it is impossible to calculate (given infinite resources) the targeted information.

More formally, we define single-server PIR as follows:

Definition 1.1. *Single-Server PIR Problem: There is a server which possesses a database M composed of n elements. A requester wishes to access element m_{i_0} from the database. The requester sends a query V to the server and the server computes a response r which is returned. The server must not be able to determine the index k given V and the requester must be able to determine the value of m_k from r.*

The scheme proposed by Aguilar-Melchor and Gaborit, which we will refer to as the Aguilar-Melchor Gaborit Scheme, is a single-server scheme and is an example of computational PIR.

A very simple (the "trivial scheme") single-server PIR protocol is to simply have the server return the entirety of the database at every request. This is information theoretically

secure, but it requires a large amount of communication to be sent upon every request. In an attempt to reduce communication requirements, computation-based proposals have been suggested. Lipmaa [3] and Gentry and Ramzan [1] provide two such schemes against which Aguilar-Melchor and Gaborit compare their protocol. These schemes— along with most if not all others—are based on number-theoretic cryptographic primitives, and are among the most practical of these. Compared to the trivial solution, these schemes greatly reduce the the amount of communication needed but at great computational cost.

The Aguilar-Melchor Gaborit Scheme is a a proposed alternative scheme which provides middle ground between the computation-heavy number theoretic schemes and the communication-heavy information-theoretic solutions. This scheme, the authors hope, will provide a more practical PIR protocol, aiming for moderate computation, communication, and time costs. However, little is known about the security of this scheme. In the original paper, the authors discussed the connection between the security of this scheme and an NP-Hard problem, but could not show a reduction. The paper also discussed reasons why lattice attacks were unlikely to work, yet the security of the scheme is not well understood, prompting further research into the matter.

Our goals for this paper were to attempt to make progress on any of the following regarding the security of the Aguilar-Melchor Gaborit Scheme:

1. Prove the NP-Hardness of the computational problem behind their scheme.
2. Apply lattice attacks and techniques from related work to break or weaken the scheme.
3. Consider the proposed parameters of the scheme, and attempt to see if any optimizations can be performed or show that the proposed parameters are insufficient for security.

Note that, as with the knapsack-based cryptosystems, we can both reduce the scheme in question to a hard problem, which proves that in the worst case the underlying problem is difficult, and yet also devise a lattice-based attack, which demonstrates that the problem is not as difficult in the average case.

We will organize our paper as follows. In Section 3, we will discuss the scheme proposed by Aguilar-Melchor and Gaborit, with an emphasis on query formation. Then we will discuss in Section 4 the lattice-based problems upon which the protocol is based, the Hidden Lattice Problem and the similar Differential Hidden Lattice Problem, and their relation to known hard problems. In Section 2, we introduce related work which will be of relevance for analyzing the security of this scheme. Finally, we discuss our attempts to determine security properties of the scheme in Section 5.

2 Related Work

We looked to relevant research for help in analyzing the security of the Aguilar-Melchor Gaborit Scheme. The first source we examined was Nguyen and Stern's overview of the use of lattices in cryptology [6] which summarizes the algorithmic and theoretic work surrounding lattices, the use of lattices to break cryptographic schemes, and the attempted use of lattices to construct cryptographic schemes. This paper provided useful breadth and understanding of lattice topics mentioned briefly in class, and gave us a good starting

point for this project. However, as we delved deeper into the PIR scheme at hand, we realized that it was less possibly relevant than we originally believed.

Additionally, we examined the attack on the Goldreich-Goldwasser-Halevi public key system [5] as the attack also involves analyzing noise and using lattice attacks. The attack was performed by first noting that the nature of the noise ensured that information was leaked about the message vector m. This was mainly due to the fact that the noise was of the form $\{\pm\sigma\}^n$ for some selected σ. This allows one to find the value of $m \mod 2\sigma$. From here, one can utilize this information to convert the problem of finding the message to a relatively easy instance of the Closest Vector Problem (CVP).

3 Aguilar-Melchor Gaborit Scheme

This scheme is composed of three separate protocols:

- Query Formation: a requester computes the query to be sent to the database, obfuscating the target in the process
- Server Response: the computation done by the database, which must correctly encode the elements of the database such that the server cannot determine what the target of the query was but the requester will be able to reverse the obfuscation)
- Information Extraction: the requester's final stage, wherein the server response is decoded into the desired database element by using information known only to the requester

3.1 A High-Level Discussion

The security of the scheme is dependent on the server being unable to distinguish the target element given generated query, so the focus of our cryptanalysis will lie with the results of the first protocol (Query Formation), as the query comprises the information that the server has access to. The main importance of the other two protocols is in proving the correctness of the scheme itself, which is shown in the original paper (and discussed here)–after all, if the requester cannot decipher the server's response, the scheme is as good as useless.

The heart of this protocol lies in the addition of controlled noise in the form of randomly generated "soft noise" matrices (with elements in $\{\pm 1\}$) and a single "hard noise" matrix (a soft noise matrix, with elements along the diagonal multiplied by q). The scheme proposes that a hard noise matrix (indicating the target of the query) is indistinguishable from a set of soft noise matrices after a series of obfuscastion techniques corresponding to randomly generated elements. The server, which does not have this private information, is unable to discover the target of the query. However, the requester is able to use the knowledge of its randomly chosen matrices—as well as the property that $q \gg 1$—to decipher the server response and recover the desired information.

To generate a query, the requester produces an ordered set of n matrices (one for each element in the database), of dimension $N \times 2N$. These matrices share a common basis M of some lattice, and each have a controlled amount of noise added to the right-hand half of the matrix (ie. a noise matrix $D_i\Delta$ is added to a matrix M_i' by computing $M_i' + [0|D_i\Delta]$). $n-1$ of these matrices have a small amount of noise added ("soft noise matrix", or SDM)

and the final matrix, which is the target of the query and corresponds to the i_0th element, has a larger amount of noise added ("hard noise matrix", or HDM). Then, a common, random permutation is applied to each matrix to form the final set of matrices sent in the query.

3.2 The Protocols

The scheme contains some global integer parameters. The database is formed of n l-bit elements, and we denote with i_0 the element in the database which we wish to query for.

Each element in the database will further be split into N sub-elements during the server response. (Note that the dimension of the lattice is $2N$.) Each sub-element is an integer can be represented in l_0 bits, where $l_0 = \lceil \log(n \times N) \rceil + 1$.

Below, we present the three protocols defined in the Aguilar-Melchor Gaborit Scheme.

3.2.1 Query Formation

1. Fix query parameters $q = 2^{l_0}$, and p some prime such that $p > 2^{3l_0}$.
2. Generate two $N \times N$ matrices in Z/pZ, A and B, such that A is invertible. Create the $N \times 2N$ matrix $M = [A|B]$
3. Generate the $N \times N$ scrambling matrix $\Delta \in Z/pZ$ as a random diagonal matrix. This will be applied to both the hard and soft noise matrices to obfuscate them and disguise which type of noise belongs to each matrix.
4. For each $i \in \{1, \ldots, n\}$, generate a random $N \times N$ matrix $P_i \in Z/pZ$ and compute the matrix $M_i'' = P_i M = [A_i|B_i]$
5. For each $M_i'' = [A_i|B_i]$, compute M_i' by adding to B_i some scrambled noise matrix $D_i\Delta$. D_i is hard noise for the target of the query and soft noise otherwise.
   - For each $i \in \{1, \ldots, n\} \setminus i_0$ (that is, for every element other than the query target), generate an $N \times N$ soft noise matrix D_i over $\{-1, 1\}$. Compute the softly disturbed matrix (SDM) by adding $D_i\Delta$ to the right half of the matrix M_i''. Thus, let $M_i' = M_i'' + [0|D_i\Delta] = [A_i|B_i + D_i\Delta]$.
   - For i_0, the target of the query, generate the hard disturbed matrix D_{i_0} by generating a soft noise matrix and replacing each element along the diagonal with q. Then compute the hard disturbed matrix (HDM) $M_{i_0}' = M_{i_0}'' + [0|D_{i_0}\Delta] = [A_i|B_i + D_i\Delta]$
6. Choose a random column permutation $\mathcal{P}(\cdot)$ and apply it to each matrix, ie., for each $i \in \{1, \ldots, n\}$, compute $M_i = \mathcal{P}(M_i')$.
7. Send the ordered set $\{M_1, \ldots, M_n\}$ as the query to the database.

We believe that the security of this scheme lies in its random generation. Because of the scrambling matrix Δ, there is no easily discernable linear relationship between soft and hard noise matrices that are used to create noise in the query. The random permutation, along with multiplication of the original basis M with randomly generated invertible P_i, renders the server unable to distinguish which columns of the matrix were disturbed with noise (this is similar to the Hidden Lattice Problem, which is further discussed in Section 4.1.

3.2.2 Server Response

The server receives the set $\{M_1, \ldots, M_n\}$ and constructs the response vector V of length $2N$.

1. Split each database element m_i into a vector of N l_0-bit integers, $\{m_{i1}, \ldots, m_{iN}\}$.
2. For each vector m_i, construct the response $v_i = \sum_{j=1}^{N} m_{ij} M_{ij}$. Each jth element in the vector m_i is multiplied by the jth row of the matrix M_i in the query vector.
3. Construct V by summing each element vector, $V = \sum_{j=1}^{N} v_j$.

3.2.3 Information Extraction

Upon recieving the server response V, the requester is then able to extract the hard noise and determine the element of interest m_k. This can be done using the values of $\mathcal{P}$, Δ, A, B, and q decided on in the query formation.

1. Compute the non-permuted noisy vector $V' = \mathcal{P}^{-1}(V)$.
2. Retrieve $S = V_D - V_U A^{-1} B$, the scrambled noise, V_U and V_D being the undisturbed and disturbed halves of V.
3. Compute the unscrambled noise: $E = S\Delta^{-1}$.
4. For each e_j in $E = [e_1, \cdots e_N]$, round e_j to the nearest multiple of q giving us values $E' = [e'_1, \cdots e'_N]$.
5. For each $j \in \{1, \cdots, N\}$, compute $m_{i_0 j} = e'_j q^{-1}$.

3.3 Correctness

This PIR scheme is built upon the fact that the hard noise is extractable from the server response. As the authors note, each element e_j will have the value

$$e_j = m_{kj} q + \sum_{i \in \{1 \cdots n\} \setminus i_0} \sum_{k=1}^{N} m_{i_0 k} \left(D_i\right)_{jk}$$

It is important to note that each element l_0-bit integer m_{ij} is clearly bounded by 2^{l_0}, and each element in any soft noise matrix is ± 1. Additionally, the choice of a value for l_0 implies that $2^{l_0 - 1} > n \cdot N$. Hence the absolute value of the summation on the right is upper bounded by

$$\sum_{i \in \{1 \cdots n\} \setminus i_0} \sum_{l=1}^{N} 2^{l_0} \left(D_i\right)_{jk} \leq n \cdot N \cdot 2^{l_0} < 2^{l_0 - 1} \cdot 2^{l_0} = 2^{2l_0 - 1} = q/2$$

Hence, the soft noise is less than half of q, so rounding to the nearest value of q will result in the $m_{i_0 j} q$. This value is less than p, which allows the final step of the last protocol to extract $m_{i_0 j}$, $e'_j q^{-1} = m_{i_0 j} q q^{-1} = m_{i_0 j}$, proving the correctness of this scheme.

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 1: Performance Analysis

3.4 Performance

Fig. 1 is taken directly from Aguilar-Melchor and Gaborit's original paper [4]. The comparison is based on the query and download for a 3Mb file from a database with 1000 elements.

One of the major advantages of this PIR scheme over previously existing schemes is its performance. Aguilar-Melchor and Gaborit compare their PIR scheme to two previously existing single-server PIR schemes (Limpaa's and Gentry and Ramzan's) with regard to generated query size, query generation time, and server download time. While the Aguilar-Melchor Gaborit Scheme must transmit a relatively much greater amount of data during its protocols in order to complete a query (resulting in a larger and slower query), the computational performance of the server response protocol mean that this scheme might actually be practical, spending a total of just 15 minutes on the transfer rather than more than a 17 hours.

It is also clear that computation is the bottleneck in all of the above schemes. The bandwidth usage never reaches above 1.2% in even the Aguilar-Melchor Gaborit Scheme, showing that the communication overhead is low compared to modern available speeds. Additionally, the use of recursion (that is, dividing a single database server into smaller, virtual servers; instead of sending a single query with the index of the targeted element, the requester sends multiple queries, which indicate the indices of the desired virtual server, then finally the local element in the smaller server) in the Aguilar-Melchor Gaborit Scheme reduces the size of its query significantly, while not changing the download time.

Aguilar-Melchor and Gaborit do not include server response computation time or response size, and do not specify what is included in the server's "Download time".

4 Equivalent Security Problems

The original paper on the Aguilar-Melchor Gaborit Scheme introduces two lattice-based problems: the Hidden Lattice Problem (HLP) and the Differential Hidden Lattice Problem (DHLP). These problems are defined to investigate the structural security of the Aguilar-Melchor Gaborit Scheme; by comparing the problems to each other, to the Aguilar-Melchor Gaborit Scheme, and to a known NP-Complete problem, the authors provide some measure of assurance that an intelligent, computationally-bounded adversary will be unable to obtain undesired information about a client's queries to the database. Below we restate the two problems from the original paper (with minor alterations for improved clarity) and demonstrate how they relate to the Aguilar-Melchor Gaborit Scheme.

4.1 Hidden Lattice Problem

Construction:

1. Define a vector space V of dimension k and of length n (such that $n > k$) over a finite field $GF(p)$.
2. Construct r random basis for V, denoted as $\{V_1, \cdots, V_r\}$. Denote the j^{th} column of V_i as $V_{i,j}$.
3. Select some subset of columns S' such that S' contains k linearly independent vectors for every basis V_i. [1] Choose a nonempty subset of size w values from the set of columns not in S', $\{1, \cdots, r\} - S'$. Denote this set as $S = \{s_1, \cdots, s_w\}$.
4. $\forall i \in \{1, \cdots, r\}, \forall s \in S$: Let $q_{i,s}$ be a random element in $GF(p)$. Construct a column $C_{i,s}$ from elements in $\{q_{i,s}, -q_{i,s}\}$. Define C_i as the set of columns for i.
5. Randomly select some $i_0 \in \{i, \cdots, r\}$ and some $q_{hard} \in GF(p)$ where $1 \ll q_{hard} \ll p$.
6. For every $g \in \{1, \cdots, w\}$, multiply the g^{th} coordinate of $C_{i_0,g}$ by q_{hard}.
7. Let U be the set of basis equal to V. Add the values of each $C_{i,s}$ to $U_{i,s}$.

Problem: Given U, determine S; that is, given the set of disturbed basis, determine which set of columns have been disturbed by the addition of noise.

Relationship between HLP and AMG scheme: The Aguilar-Melchor Gaborit Scheme is an instance of the Hidden Lattice Problem with the following parameters. The dimension of V equals the number of disturbed columns, which equals the number of bitstrings that each element in the database is separated into ($k = w = N$). The size of V is equal to two times the number of sub-elements that each element in the database is divided into upon Server Response ($n = 2N$). $q_{hard} = q$, the global hard noise parameter. C_{i_0} represents the hard noise matrix after multiplying it by the scrambling matrix ($D_{i_0}\Delta$). Every other C_i represents the columns of a soft noise matrix after multiplying by the scrambling matrix ($D_i\Delta$). The random selection of columns in S for the HLP is similar to the permutation of columns in the original scheme. Thus an algorithm that is capable of solving the Hidden Lattice Problem is also capable of determining which columns were disturbed among the query matrices.

Theorem 1. *If an adversary knows the columns with noise, then they can determine which matrix among the query set is the HDM.*

Recall that $M = [A|B]$ and for all $0 < i < n$, $M_i = P_iM = [P_iA|P_iB] + [0|D_i\Delta]$. We omit the permutation, which is negated by the adversary's knowledge of the hidden lattice. Suppose the adversary knows which columns have noise added (without loss of generality, say these are the first N columns).

The following two algorithms shows how to recover the values of all $D_i\Delta$ given an adversary that has solved the HLP. It comes from a generalization of Lemma 1 and analysis that follows, taken from the Aguilar-Melchor and Gaborit's paper of interest.

[1] We assume such a selection of columns exists without loss of generality. If such a grouping of columns did not exist we could rearrange the columns of each basis V_i during generation until a sufficient selection of columns existed.

Algorithm 1. *Solving for One Column (In the Noise Matrices)*

for $i \leftarrow 1$ to n **do**
  for $j \leftarrow (i+1)$ to n **do**
    First, using Lemma 1 introduced in Aguilar-Melchor and Gaborit's paper, the adversary can compute $P_j P_i^{-1}$. Since we know that the first N columns are unmodified, we can determine through inspection the values of $P_i A$ and $P_j A$. Then simply evaluate $P_j A (P_i A)^{-1} = P_j A P_i A^{-1} A^{-1} = P_j P_i^{-1}$.
    The adversary then computes $S_{ij} = P_j P_i^{-1} M_i - M_j = P_j P_i^{-1}(P_i M + D_i \Delta) - (P_j M + D_j \Delta) = [0|P_j P_i^{-1} D_i \Delta] - [0|D_j \Delta]$.
    Create a set of N linear equations by using the elements of the (unknown) cth columns of $D_i\Delta$ and $D_j\Delta$ and the (known) cth column of S_{ij}.
  end for
end for
At the end, there should exist nN equations and nN unknowns. This can be solved in $O((nN)^3)$ time, obtaining the values of the cth columns of all $D_i\Delta$.

Algorithm 2. *Solving for All Disturbed Matrices*

for $c \leftarrow 1$ to $2N$ **do**
  Use Alg. 1 to solve for the values of the cth column of $D_i\Delta$.
end for

At this stage, the adversary has the complete values of $D_i\Delta$ for all $i \in \{1, \ldots, n\}$. Since all added soft noise is of the form $\{-1, +1\}$, all SDMs are easily relatable to Δ. There should be only a single matrix in the set for the query that differs from this pattern, which will be distinguishable from the HDM.

It is not necessary for the hard noise values (originally manifested as q along the diagonal of the HDM) to lie in a particular pattern in the matrix, as the HDM will be the only matrix which contains elements not in Δ. Thus, it is unnecessary for an attacker to exactly reverse the permutation; it is enough to find the disturbed columns in order to distinguish the index of the HDM.

This algorithm (which the authors admit is naive in construction) will run in time $O((n^3N^4)$. The inner loop (pairwise equation generation for each of n matrices) is dominated by the $O((nN)^3)$ matrix inversion used to solve the system of equations, and must be repeated for each of $2N$ columsn in the lattice.

An adversary that is able to distinguish the unmodified columns of the basis that make up a query would prove the Aguilar-Melchor Gaborit Scheme insecure, but showing that an adversary is unable to do so is insufficient for proving the security of the scheme. To break the Aguilar-Melchor Gaborit Scheme, an adversary need only distinguish which of the basis corresponds to i_0 (that is, the one basis that corresponds to the element in the database that the client is requesting in a given query). This is equivalent to distinguishing which basis has been disturbed with a hard noise matrix from those disturbed with soft noise matrices. Aguilar-Melchor and Gaborit introduce a more specific problem, the Differential Hidden Lattice Problem, to represent this concept.

4.2 Differential Hidden Lattice Problem

Construction:

1. Define a a vector space V of dimension k and of length n (such that $n > k$) over a finite field $GF(p)$..
2. Construct r random basis for V, denoted as $\{V_1, \cdots, V_r\}$. Denote the j^{th} column of V_i as $V_{i,j}$.
3. Define T_1 and T_2 to be two subsets of $\{1, \cdots, r\}$ such that $T_1 \neq T_2$
4. Select some subset of columns S' such that S' contains k linearly independent vectors for every basis V_i. Choose a nonempty subset of size w values from the set of columns not in S', $\{1, \cdots, r\} - S'$. Denote this set as $S = \{s_1, \cdots, s_w\}$.
5. $\forall i \in \{1, \cdots, r\}, \forall s \in S$: Let $q_{i,s}$ be a random element in $GF(p)$. Construct a column $C_{i,s}$ from elements in $\{q_{i,s}, -q_{i,s}\}$. Define C_i as the set of columns for i.
6. Randomly select some $d \in \{1, 2\}$ and some $q_{hard} \in GF(p)$ where $1 << q_{hard} << p$. Let $T = T_d$.
7. For every $g \in \{1, \cdots, w\}$and every $f \in T$, multiply the g^{th} coordinate of $C_{f,g}$ by q_{hard}.
8. Let U be the set of basis equal to V. Add the values of each $C_{i,s}$ to $U_{i,s}$.

Problem: Given T_1, T_2, and U, determine the value of d. That is, given two possible sets of basis and the set of disturbed basis, distinguish which of the two sets of basis was given the addition of hard noise.

Relationship between DHLP and AMG scheme: The relationship between DHLP and the AMG scheme is similar to the relationship between HLP and the AMG sceme in terms of equivalent parameters. DHLP provides two possible subsets of basis that are disturbed with hard noise and requires an adversary to distinguish which one was used to generate a set of disturbed basis. The Aguilar-Melchor Gaborit Scheme corresponds to a similar case where only one basis is disturbed. Thus if an adversary were able to solve DHLP they would be able to distinguish which basis was disturbed with hard noise and determine which element in the database a client was asking for, breaking the security of Aguilar-Melchor Gaborit Scheme.

Relationship between HLP and DHLP: As Aguilar-Melchor and Gaborit claim, HLP is at least as hard as DHLP. An adversary capable of solving HLP is also capable of solving DHLP. As shown Theorem 1 an adversary capable of solving HLP is able to obtain every matrix $D_i\Delta$, and can then determine which of these matrices have been disturbed by the multiplication of q_{hard}.

5 Security Analysis

Here, we attempt to build upon the authors' analysis of the security of their scheme. Our main goals in this section are to reduce an NP-Hard problem to the computational problems underlying the security of the Aguilar-Melchor Gaborit Scheme or to show that

it can be attacked through methods such as lattices. We have not been successful in doing either, but we will give an overview of our attempts to do so through analyzing related work in the areas of coding theory, noise, and lattice attacks.

5.1 Reduction to a Hard Problem

Aguilar-Melchor and Gaborit state that the Hidden Lattice Problem is "very likely hard" by observing its similarities to the Code Puncture Search Problem (CPSP), a variant of the known NP-Complete Punctured Code Problem. We take Aguilar-Melchor and Gaborit's justification a step further by providing a reduction between CPSP and HLP by showing that an instance of CPSP is actually a special case of HLP, and a black box solution of HLP can thus be used to solve CPSP. Below is the definition of CPSP as defined in [4]:

Code Puncture Search Problem: Let H be a $k \times m$ matrix of rank k and M be 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 matreices (H and M) which are the s random columns of M.

An instance of CPSP is an instance of HLP: [2] Let r in HLP (the number of basis to be disturbed) be equal to 1. If $|V| = r = 1$, we know that $i_0 = 1$ as there is only one matrix available to recieve the addition of hard noise columns (columns disturbed by the multiplication of q and one element in each column). Let q be smaller than every element in M, and p be a prime number sufficiently large such that no element ever exceeds p. Consider the first random column added to M, $s = [s_1|s_2|\cdots|s_k]$. Construct the vector $s' = [s_1 - q|s_2 - 1|\cdots|s_k - 1]$. This gives $s = s' + [q|1|\cdots|1]$. s corresponds to a disturbed column in the result of HLP, where $r_j = 1$ for the soft noise addition and $q = q$ for the hard noise addition. This generalizes; all columns randomly added in the instance of the CPSP can be interpreted as columns of the HLP after being disturbed by soft noise equal to 1 and a hard noise coordinate equal to some value q.

HLP is NP-Hard. Assume that an algorithm that solves the Hidden Lattice Problem in polynomial time exists. Provide this algorithm an instance M of the Code Puncture Search Problem as input. As we have demonstrated above, M can be interpreted as an instance of HLP where the randomly added columns of M correspond to disturbed columns in the HLP. Running the HLP algorithm returns this set of columns, which can then be returned as the solution to CPSP. We have successfully solved CPSP in polynomial time. However, CPSP is proven to be NP-complete. Thus HLP is NP-Hard, as it is at least as hard as CPSP.

HLP is verifiable in polynomial time. Theorem 1 in Section 4.1 demonstrates how an adversary with knowledge of the location of permuted columns can construct the matrices $D_i\Delta$ for all i in polynomial time. A verifier runs this and verifies that, for all but one $D_i\Delta$, each column is composed of the same element (positive or negative), and that for one $D_{i_0}\Delta$ all but one element of each column are are same (again positive or negative), and that the differing element differs from the rest of the elements of the column by some factor q.

Note that in Section 4.1 we show the Aguilar-Melchor Gaborit Scheme is a special case of HLP. It is possible that this special case can be solved more efficiently. However, we believe it to be unlikely, and that any break in the security of the Aguilar-Melchor Gaborit

[2] In the course of our research this was one of the proposed reductions. As a group we aren't currently sure of its correctness, but it's what we've got.

Scheme is unlikely to come from the discovery of the permutation at the end of Protocol 1. This result also says nothing of the hardness of the Differential Hidden Lattice Problem; we showed that HLP is at least as hard as DHLP but have not proven the other direction. The hardness of DHLP is, as far as these authors are aware, still an open question.

5.2 Lattice-Based Attacks

We begin this section by giving an overview of lattice and their use in breaking cryptographic schemes before summarizing the authors' remarks on why lattice attacks do not appear to be of use on their PIR scheme. Then we will introduce related work and explain our own attempts to break this scheme.

Remember that the Aguilar-Melchor Gaborit Scheme is broken if and only if we can determine which of the matrices is the hard-noise matrix among the query consisting of n $N \times 2N$ matrices $\{M_1, \cdots, M_n\}$. As shown in section 3, this problem is equivalent to determining which of the columns in the matrices of the query contain noise and which are unmodified.

While breaking this scheme is quite likely hard in the worst case, it may be breakable in practice by lattice attacks. Lattices are discrete sub-groups of $\mathbb{R}^n$ - or in this example $GF(p)^n$. The lattice problems of Shortest Vector Problem (SVP) and Closest Vector Problem (CVP) are known to be hard, but can often be solved relatively efficiently using the LLL algorithm. One can utilize these properties to use lattice attacks to solve similar hard problems. A lattice attack consists of finding an analogy between the solution to the problem we are attempting to solve and a closest or shortest vector problem over a lattice. After determining the basis of the lattice, the LLL algorithm can be utilized to solve the problem in a reasonable amount of time.

There is a seemingly similar problem which can be attacked using lattices. Note that these matrixes are over $GF(p)$ for some prime p.

Theorem 2. *Let M be a $N \times 2N$ matrix. Let $M' = HM + R$ where H is an invertible $N \times N$ matrix and R is a $N \times 2N$ matrix with N columns containing 0's and N columns of noise of the form $\{\pm 1\}^N$. Given M and M', one can determine which columns of M' contain noise.*

Without going into the low-level details, the lattice method used to determine the noisy columns in M' works by finding the noise in a single row of M'. The noise in any row will consist of a vector of length $2N$ with N 0's and N elements ± 1. This is a fairly short vector, and an analogy can be made between the problem of finding this noise vector and the SVP problem in a properly chosen lattice. Given the noise in a row, the columns of noise are clearly the indices of non-zero elements of the row.

However, there are several notable differences concerning the noise introduced to these columns and the hard and soft noise introduced into the matrices in the query formation of the scheme in question. The noise in a row of any M_i in the query will consist of N 0's and N random non-zero elements in $GF(p)$ thanks to the noise-scarmbling matrix Δ. he chance of the noise being a shortest vector is negligible. Additionally, in the above example, the hidden lattice and the disturbed lattice are presented. When trying to break a query, we only have access to various disturbed lattices constructed from the same hidden lattice. This certainly means that breaking the scheme in question will require a different approach (although whether it will be harder is easier is unclear).

Finally, our results from comparing the security scheme to the related work described above also failed. Notably, our examination of the breaking of the GGH scheme [5] also relied on the predictable structure of the added noise. The noise values are randomly chosen in the scheme in question, so once again we doubt this attack will be of help.

Given the examples above, it would intuitively seem that this scheme is safe from lattice attacks and is secure. However, an important thing to note in this example is the relatively small dimention of the lattice. The dimension is $2N$ for a chosen parameter N. The authors recommend the value $N = 50$ as it would require a brute force approach to require over 2^{100} calculations. Still, such a low dimension seems succeptable when compared to the much larger dimensions used in many other Lattice based schemes. For example, Goldreich, Goldwasser, and Halevi [2] only claimed that their attacks on their system would be beyond a possibility of attack at dimension 300. At dimension 100, their scheme was breakable with non-negligble probability. This cryptosystem and others will require more analysis, but for now, the choosing of a parameter $N = 50$ and thus dimension 100 seems suspicious.

5.3 Parameter Analysis

It is entirely possible to create a scheme which is theoretically secure while being vulnerable to brute-force attacks due to small parameters used in computation. This may render the scheme unusable if, for example, using larger parameters result in unreasonably long (but still polynomial) computation times to generate queries or responses. It is always important to keep in mind a practical application of the scheme rather than simply theoretical bounds.

The paper suggests the following values for parameters:

- N: number of sub-elements each database element is split into. Suggested: $N = 50$
- l_0: number of bits each sub-element contains. Suggested: $l_0 = 20$. From this, $q = 2^{2l_0} = 2^{40}$. This allows for a value of n up to $2^{l_0-1}/N \approx 1 \times 10^4$.

An analysis of possible lattice-based attacks to characterize solutions is discussed above (Section 5.2). Another attack is a brute force attack on the Hidden Lattice Problem; that is, the server can guess a subset of N columns that it believes are added noise columns. If the server correctly identifies the noise columns in the query, it can compute a basis of the Hidden Lattice Problem.

There are $\binom{2N}{N}$ subsets to choose from. For the suggested $N = 50$, there exist a total of $\binom{100}{50} \approx 1 \times 10^{29}$ subsets. Theorem 1 uses a brute force algorithm (Alg. 2) in order to solve for the values of $D_i\Delta$, which takes time $O(n^3N^4)$ in order to do the full computation for a given subset.

One might expect to guess, on average, about 50% of the subsets before hitting the correct choice. This gives you just 5×10^{28}. Even assuming that each subset guess would take only a millisecond to compute, this would take a little over the age of the universe to brute force (unless I did something totally embarassingly wrong with that math).

However, in many cases, it won't be necessary to compute the values of the entirety of each $D_i\Delta$ matrix to rule out a subset from being a solution of HLP. As an adversary computes each column of values across the n noise matrices (ie. after applying Alg. 1 for a given column), the expected form is $n - 1$ matrices with integers equal to $\pm x$ (for some

$x \in GF(p)$, and one matrix with an integer equal to $qx \mod p$. Any column that does not follow this pattern immediately removes the current subset from being a possible solution of HLP. Assuming (without proof, because I don't know how to prove this) it is a rare occurrence for any invalid subset to follow this pattern (and doubly so for it to occur more than once), this reduces the naive runtime approximately by a factor of $2N$—a hopeful but ultimately fruitless gain given the rest of the runtime—yielding an overall runtime of $O((nN)^3)$ per attempt. Unfortunately, that doesn't make much of an impact on this dumb attack. To paraphrase the authors (albeit on a slightly different topic), this isn't quite an attack but *perhaps* something that might play a part in one.

6 Further Work

There is still much work to be done regarding this scheme. Much is left unsaid about its security. We have proposed a reduction from Hidden Lattice Problem to a known NP-Complete problem and determined that it appears hard to determine permutation of the columns in the query matrices, but have not yet classified the hardness of the Differential Hidden Lattice Problem. Proving that DHLP is hard would be necessary in demonstrating that Aguilar-Melchor Gaborit Scheme is provably secure (and proving DHLP to be easy would break the scheme completely).

It is unknown as of yet if lattice attacks are plausible. As posited by the authors, it is difficult to characterize some relationship between hard noise and soft noise matrices, which are key in attempting to break this cryptosystem. The addition of randomizing noise in addition to structured, controlled noise renders many previously-used paradigms incompatible with this scheme.

Investigation into the suggested parameters showed that they are secure against brute force attacks, yet there is always the chance that an adversary more clever than us will devise some way of showing that they are insufficient. It may come in the form of a lattice-based attack, or simply take advantage of some quirk of the PIR scheme.

References

[1] Craig Gentry and Zulfikar Ramzan. Single-database private information retrieval with constant communication rate. In *In Proceedings of the 32nd International Colloquium on Automata, Languages and Programming*, pages 803–815. Springer-Verlag, 2005.

[2] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. pages 112–131. Springer-Verlag, 1996.

[3] Helger Lipmaa. An oblivious transfer protocol with log-squared communication. In *In Proceedings of the 8th International Conference on Information Security*, pages 314–328. Springer-Verlag, 2005.

[4] Carlos Aguilar Melchor and Philippe Gaborit. A lattice-based computationally-efficient private information retrieval protocol. *IACR Cryptology ePrint Archive*, 2007:446, 2007.

[5] Phong Nguyen. Cryptanalysis of the goldreich-goldwasser-halevi cryptosystem from crypto '97. In *In Proc. of Crypto '99, volume 1666 of LNCS*, pages 288–304. Springer-Verlag, 1999.

[6] Phong Q. Nguyen and Jacques Stern. The two faces of lattices in cryptology. In *Revised Papers from the International Conference on Cryptography and Lattices*, CaLC '01, pages 146–180, London, UK, UK, 2001. Springer-Verlag.