IACR Communications in Cryptology
ISSN 3006-5496, Vol. 3, No. 1, 23 pages.

https://doi.org/10.62056/an-4c3tw9

Cryptanalysis of a Lattice-Based PIR Scheme for Arbitrary Database Sizes

Svenja Lage

German Aerospace Center (DLR), Institute of Communications and Navigation, Germany

Abstract. Private information retrieval schemes enable users to privately retrieve files from a server without disclosing the content of their queries. In 2008, Aguilar Melchor and Gaborit proposed a PIR scheme that balances communication and server-side computational cost. For particularly small databases, Liu and Bi identified a vulnerability in the scheme using lattice-based methods. Nevertheless, the rapid increase in computational cost associated with the attack limited its practical applicability, leaving the scheme's overall security largely intact. In this paper, we present a novel two-stage attack that extends the work of Liu and Bi to databases of arbitrary sizes. To this end, we employ an innovative binary-search-like preprocessing technique, which enables a significant reduction in the number of lattice problems that need to be considered. We demonstrate how to compromise the scheme in a matter of minutes using an ordinary laptop. Our findings are substantiated through both rigorous analytical proofs and comprehensive numerical experiments.

Keywords: private information retrieval · lattice-based cryptography · cryptanalysis

1 Introduction

Private Information Retrieval (PIR) schemes enable users to access data from a server while preventing the server from learning the specific file being requested. Thus, PIR schemes maintain the confidentiality of the user's queries. However, this process introduces significant computational overhead. In the worst-case scenario, the user would have to download the entire database to securely conceal the query. Although the request remains hidden, this solution is impractical for large databases with many files. Notably, as demonstrated in [CKGS98], this is the only method to achieve information-theoretical security within a single-server scenario.

As a more practical alternative, the concept of computationally secure PIR (cPIR) schemes has emerged, providing a trade-off between privacy and communication cost. Built on traditional cryptographic techniques, early cPIR schemes were mostly founded on number-theoretic problems, such as integer factorization (e.g., see [KO97], [CMS99]). However, these schemes were subsequently shown to be computationally inefficient [SC07] and, moreover, they are not resistant against potential attacks by quantum computers [Sho97]. Motivated by the need for post-quantum security, modern PIR schemes predominantly rely on hard problems in lattice theory, which are believed to be resistant to quantum computer attacks (compare for example [HHC+23], [AMBFK16] or the generic transformation from homomorphic encryption in [BV11]). Recently, code-based constructions have also gained attention as a promising alternative for achieving quantum security in PIR schemes ([HHWZ20], [HV25]).

E-mail: svenja.lage@dlr.de (Svenja Lage)

Received: 2026-01-23 Accepted: 2026-04-21

The majority of PIR schemes focus on optimizing communication cost while ensuring a predetermined level of security. In 2008, Aguilar Melchor and Gaborit [AMG08] introduced a PIR scheme that addressed both communication costs and the often-overlooked computational burden on the server side. The authors aimed at balancing both aspects to develop a secure and efficient PIR scheme. Although more recent and efficient PIR schemes have been proposed, the PIR scheme presented in [AMG08] remains a foundational framework for the development of new PIR constructions. This influence is evident in code-based schemes such as [HHWZ20] and [HV25]. In 2016, Liu and Bi [LB16] discovered a linear relationship between the queries and the secret noise matrix holding the information about the desired file index, which can be exploited using a lattice reduction approach. However, their attack requires performing an amount of lattice reductions which grows linearly with the number of files in the database, rendering it infeasible for large databases and hence leaving the overall security of the scheme intact.

In this paper, we introduce a two-stage attack that effectively compromises the scheme for databases of all sizes. Our approach entails a novel preprocessing step, where we presort the files under consideration using an innovative binary-search-like method. This preprocessing step enables the former attack to be applied on only a small subset of files, which yields a significant reduction in computational complexity. The preprocessing step scales logarithmically with the number of files in the database, rendering the attack highly scalable. After introducing the original scheme and the initial attack, Section 5 describes our enhanced attack and proves its correctness. Subsequently, a numerical analysis is presented to underscore the efficiency of our attack. Notably, the binary-search-inspired attack paradigm described in this paper can be generalized to other systems and is hence of independent value.

2 Preliminaries

We first introduce some definitions and results needed for the subsequent sections. For a prime p, let $\mathbb{Z}_p$ be the finite field with p elements and denote the set of all $n \times m$ matrices over $\mathbb{Z}_p$ by $M_{n,m}(\mathbb{Z}_p)$. For the special case of square matrices, we write $M_n(\mathbb{Z}_p)$. The subset of all non-singular $n \times n$ matrices over $\mathbb{Z}_p$ is denoted by $GL_n(\mathbb{Z}_p)$. For any $A \in M_{n,m}(\mathbb{Z}_p)$, we use $A|_{[u:v,:]}$ to refer to the submatrix of A consisting of rows u through v and $A|_{[:,u:v]}$ to refer to the submatrix of A consisting of columns u through v.

A lattice $L \subset \mathbb{R}^d$ is formally defined as a discrete subgroup of $\mathbb{R}^d$. For a set of linearly independent vectors $B = [b_1, \ldots, b_n]$ with $b_i \in \mathbb{R}^d$, the lattice generated by B is defined as

$$L(B) = \{y \in \mathbb{R}^d : y = Bx \text{ for some } x \in \mathbb{Z}^n\}. \tag{1}$$

The matrix B is called a lattice basis. The numbers n and d are referred to as the rank and dimension of the lattice, respectively.
In this work, we focus on q-ary lattices, which are defined as

$$L_q(B) = \{y \in \mathbb{Z}^d : y = Bx \mod q \text{ for some } x \in \mathbb{Z}_q^n\}$$

for a fixed $q \in \mathbb{N}$ and $B \in \mathbb{Z}_q^{d \times n}$. The lattice basis in q-ary lattices is defined as the columns of the matrix $E = (B \,|\, qI_d)$, where I_d denotes the d-dimensional identity matrix. This augmented matrix construction enables the lattice to simultaneously capture the linear combinations generated by B and the modular reductions imposed by the parameter q. Utilizing the basis E, we can display the q-ary lattice as an ordinary lattice $L(E)$ as defined in (1). Conversely, every lattice L with $q\mathbb{Z}^d \subseteq L \subseteq \mathbb{Z}^d$ is a q-ary lattice.

The Shortest Vector Problem (SVP) is a central problem in lattice-theory and serves as a basis for the security of lattice-based cryptography, typically via reductions to average-case problems such as Learning With Errors and Short Integer Solution. It is defined as follows.

Definition 1 (Shortest Vector Problem (SVP))**.** Given a basis B of a lattice L and a norm $||\cdot||$, find a non-zero vector $v \in L$ such that

$$||v|| = \min_{u \in L \setminus \{0\}} ||u||.$$

The SVP seeks the shortest non-zero lattice vector with respect to a given norm. Unless otherwise specified, in the following sections, any reference to a norm, denoted as $||\cdot||$, shall be interpreted as the Euclidean norm. The computational complexity of SVP depends on the problem's dimensionality and the quality of the lattice bases provided. However, in the Euclidean norm, the SVP is at least known to be NP-hard under randomized reductions [Ajt98].

Closely related to the SVP is the Closest Vector Problem (CVP), which asks for the lattice vector nearest to a given target vector t. The CVP is known to be NP-hard in the Euclidean norm [vEB81].

Definition 2 (Closest Vector Problem (CVP))**.** Given a basis B of a lattice L, a target vector $t \notin L$ and a norm $||\cdot||$, find a vector $v \in L$ such that

$$||v - t|| = \min_{u \in L} ||u - t||.$$

In our subsequent attack, we are initially confronted with an instance of the CVP on a lattice L and target vector t. Given the existence of more efficient solvers for SVPs, we leverage Kannan's embedding technique [Kan87] to transform the CVP into a higher-dimensional SVP.

Kannan's approach starts with a short basis $B \in \mathbb{Z}^{d \times d}$ of the lattice L, where a short basis is characterized by its reduced norm and nearly orthogonal vectors. If the initial basis B does not already possess these desirable properties, a lattice reduction algorithm is utilized to transform B into a short basis, thereby optimizing its form for subsequent computations. As a second step, an embedded lattice with basis

$$\begin{pmatrix} B & t \\ 0 & M \end{pmatrix} \in \mathbb{Z}^{(d+1)\times(d+1)}$$

is constructed for an embedding factor $M \in \mathbb{N}$. By computing the shortest vector $(e, M)^T$ in the embedded lattice, one can recover a closest lattice vector to t as $t - e$. The choice of the embedding parameter M has a crucial impact on the geometry of the embedded lattice. In particular, it determines the gap between the shortest and the second shortest vector. As observed in [WAT18], if M is chosen too large, this gap becomes small, making it difficult to distinguish the shortest vector form other short vectors. On the other hand, if M is chosen too small, the embedding may fail to encode the correct closest vector. For an appropriate choice of M, the embedded lattice exhibits a unique shortest vector corresponding to the the solution of the CVP instance, efficiently yielding an instance of the unique Shortest Vector problem (uSVP) (see e.g. [MG02]). A common heuristic, suggested in [WAT18], is to choose M proportional to the expected norm $||e||$. However, the optimal choice of M is application-dependent and typically requires a dedicated analysis.

To estimate the number of lattice points within a certain distance to a given point, we employ the following definitions. The d-dimensional sphere of radius $R > 0$ centered at $x \in \mathbb{R}^d$ is denoted by $B_R(x)$ with

$$B_R(x) = \{y \in \mathbb{R}^d : ||y - x|| \leq R\}.$$

Let $N_R(x)$ represent the number of integer points within $B_R(x)$, which is given by

$$N_R(x) = |\{y \in \mathbb{Z}^d : ||y - x|| \leq R\}|.$$

Estimating $N_R(x)$ precisely is non-trivial. However, for $d \geq 4$, it can be shown [CI95] that $N_R(x)$ can be approximated by the volume of the d-dimensional sphere as

$$N_R(x) = \frac{\pi^{\frac{d}{2}}}{\Gamma\left(\frac{d}{2}+1\right)} R^d + O(R^{d-2}),$$

where $\Gamma(\cdot)$ denotes the Gamma function and $O(\cdot)$ represents the big O notation. It is important to note that shifting the center x between integers does not change $N_R(x)$. However, shifting x to a non-integer point does indeed influence the number of integer points within the sphere, as demonstrated in [MO90].

3 The original PIR scheme

The authors of [AMG08] introduced a lattice-based PIR scheme, which operates on a database consisting of n files, each represented as an $L \times N$ matrix, $A_1, A_2, \ldots, A_n$, over $\mathbb{Z}_p$. The user aims to retrieve the file indexed by $i_0 \in \{1, 2, \ldots, n\}$, while ensuring that no information about i_0 is disclosed to the server. In this scheme, the parameter N is a crucial scheme parameter influencing the lattice dimension. Meanwhile, the parameter L is chosen to be sufficiently large to allow for the encoding of all files within the database.

3.1 Query generation

For each file in the database, the user generates a query $B_i \in M_{N,2N}(\mathbb{Z}_p)$, $i = 1, \ldots, n$, where B_i is calculated according to the following procedure.

1. Define the parameters l_0 and q as $l_0 = \lceil \log(nN) \rceil + 2$ and $q = 2^{2l_0 - 1}$, respectively. Additionally, let the prime number p be chosen such that $p > 2^{3l_0}$. In the following steps, we use q to hide the index of interest within the noise. For indices $i \neq i_0$, the noise remains small, whereas the noise corresponding to i_0 is scaled by q. Consequently, q must be large relative to the noise values. At the same time, q remains small compared to the field size p, ensuring that the file can be correctly recovered.

2. Two random matrices, $M_1 \in GL_N(\mathbb{Z}_p)$ and $M_2 \in M_N(\mathbb{Z}_p)$, are generated, along with a random scrambling matrix $\Delta \in GL_{2N}(\mathbb{Z}_p)$. These matrices will be utilized in the construction of all query matrices $B_1, B_2, \ldots, B_n$ and must be stored for later use during information extraction.

3. For each $i \in \{1, \ldots, n\}$, generate a random matrix $P_i \in GL_N(\mathbb{Z}_p)$. In addition, randomly generate noise matrices $\epsilon_i \in \{-1, 1\}^{N \times N}$ for every $i \in \{1, \ldots, n\}$.

4. For the index of interest i_0, modify the noise matrix ϵ_{i_0} by multiplying its diagonal entries with q. This step embeds the index of interest into the amplitude of the noise ϵ_{i_0}. All other matrices ϵ_i for $i \neq i_0$ generated in the previous step remain unchanged.

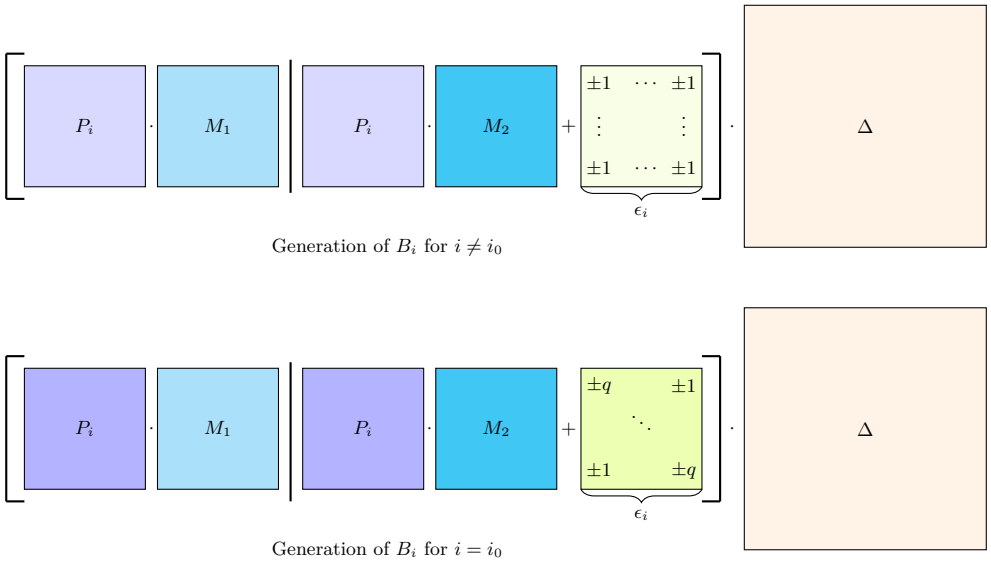

Figure 1: Generation procedure of B_i forming the query $(B_1, \ldots, B_n, p)$ in the PIR scheme of [AMG08] for $i \neq i_0$ (top) and $i = i_0$ (bottom).

5. For each $i \in \{1, \ldots, n\}$, compute the query matrix B_i as

$$B_i = [P_i M_1 \,|\, P_i M_2 + \epsilon_i]\Delta.$$

Finally, the user transmits the query $(B_1, \ldots, B_n, p)$ to the server, thereby initiating the privacy-preserving retrieval process. A graphical representation of the query generating process is shown in Figure 1. Since M_1, M_2 and Δ are used for all B_i, $i = 1, \ldots, n$, these matrices are depicted in the same color. In contrast, the matrices P_i and ϵ_i differ for each $i \in \{1, \ldots, n\}$. Furthermore, for $i = i_0$, the noise matrix ϵ_{i_0} differs from all other matrices ϵ_i with $i \neq i_0$ in that its diagonal entries are scaled by q.

3.2 Answer encoding

In the database, the files $A_1, \ldots, A_n$ are represented as $L \times N$ matrices, where each entry is an l_0-bit scalar. The server constructs the $nN \times 2N$ matrix Q from the query matrices $(B_1, \ldots, B_n)$ as

$$Q = [B_1^T | \ldots | B_n^T]^T \in M_{nN,2N}(\mathbb{Z}_p).$$

In response to the query, the server computes the $L \times 2N$ matrix R as $R = AQ$, where A is the $L \times nN$ matrix obtained by horizontally concatenating the individual files $A_1, \ldots, A_n$. Formally, A is given by

$$A = [A_1 | \ldots | A_n] \in M_{L,nN}(\mathbb{Z}_p).$$

The server performs the matrix multiplication over $\mathbb{Z}_p$ and returns R as the response to the user's query. The answer generation procedure is displayed in Figure 2.

3.3 Information extraction

Once the server transmits the query response, the user can extract the file A_{i_0} from the matrix R. To this end, the user adopts a row-wise approach. Specifically, for each row V_i, $i = 1, \ldots, L$, of R the user employs the following procedure.

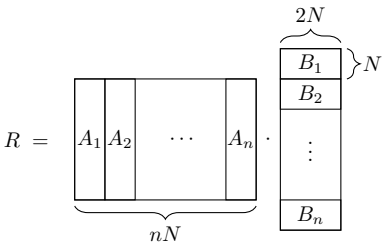

Figure 2: Answer generation within the PIR scheme described in [AMG08].

1. Since Δ is invertible, calculate Δ^{-1} and unscramble V_i as

$$V_i' = V_i \Delta^{-1}.$$

2. The user partitions the modified row V_i' into two segments: the undisturbed part $V_i'(U)$, comprising the first N columns, and the disturbed part $V_i'(D)$, consisting of the remaining N columns. Subsequently, using that M_1 is non-singular, the user computes the error vector e_i as

$$e_i = V_i'(D) - V_i'(U) M_1^{-1} M_2.$$

   It is noteworthy that the error vector e_i depends solely on the noise matrices and the database files. Furthermore, the information extraction process only requires knowledge of the matrices M_1, M_2, and Δ, thereby obviating the need for the user to store the matrices $P_1, \ldots, P_n$ at high cost.

3. For each entry $e_{i,j}$, $j = 1, \ldots, N$ in e_i set

$$e'_{i,j} = \begin{cases} p - e_{i,j} & \text{if } e_{i,j} > \frac{p}{2} \\ e_{i,j} & \text{else.} \end{cases}$$

4. Calculate

$$e''_{i,j} = \begin{cases} e'_{i,j} - (e_{i,j} \mod q) & \text{if } (e'_{i,j} \mod q) < \frac{q}{2} \\ e'_{i,j} - (e_{i,j} \mod q) + q & \text{else} \end{cases}$$

   for every $j = 1, \ldots, N$.

5. Given that the elements of the file A_{i_0} are related to the error vector $e''_{i,j}$ by $A_{i_0}[i,j] = \frac{e''_{i,j}}{q}$ for every $j = 1, \ldots, N$, the user can reconstruct the file A_{i_0} by iterating over all rows V_i of R.

Although the authors don't claim particular security levels, they analyze the scheme's vulnerability to lattice and structural attacks. For practical implementation, they finally recommend using the parameters $l_0 = 20$, $q = 2^{39}$, $N = 50$, $p = 2^{60} + 325$, and a maximum database size of $n \leq 10,000$. As intended, the authors demonstrate that the PIR scheme can be viewed as a trade-off between communication costs and computational costs.

4 Liu and Bi's attack

As demonstrated by the authors in [LB16], a linear relationship between the query $B = (B_1, \dots, B_n)^T$ and the noise matrices $\epsilon = (\epsilon_1, \dots, \epsilon_n)^T$ can be established. This linear relationship reveals a vulnerability in the scheme, particularly for small databases. Since our attack builds upon their approach, we provide a brief summary of their method.

Consider the lattice

$$L_p(B) = \{y \in \mathbb{Z}^{nN} : y = Bx \mod p \text{ for some } x \in \mathbb{Z}_p^{2N}\}.$$

The structural properties of B enabled the authors to demonstrate [LB16, Theorem 1], that the columns of ϵ are themselves contained within the lattice. This, in turn, implies the existence of a bridging matrix $D \in M_{2N\times N}(\mathbb{Z}_p)$, such that

$$\begin{pmatrix} B_1 \\ \vdots \\ B_n \end{pmatrix} D = \begin{pmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{pmatrix} \tag{2}$$

with the multiplication performed in $\mathbb{Z}_p$. The entries of the columns of ϵ are restricted to values in $\{-1, 1\}$. The only exception is the diagonal entries of ϵ_{i_0}, which take values in $\{-q, q\}$. Given that q is significantly smaller than p, the columns of ϵ can be characterized as short lattice vectors.

Initially, it may seem intuitive to apply lattice reduction techniques directly to the lattice $L_p(B)$ to search for short lattice vectors. However, a closer examination reveals that this approach is impractical due to the lattice's high dimension. Given that the complexity of lattice reduction techniques, such as LLL [LLL82], grows polynomial with the lattice dimension d, current results suggest a complexity of $O(d^4)$ [NS16], working directly on $L_p(B)$ is not feasible.

Instead of directly manipulating the original lattice, the authors of [LB16] introduce a sequence of reduced-dimension lattices, denoted by L_i, making the problem computationally manageable. Thereby, the lattices L_i are defined by

$$L_i = \{y \in \mathbb{Z}^{2N+k} : y = C_i x \mod p \text{ for some } x \in \mathbb{Z}_p^{2N}\}, \tag{3}$$

where the matrices $C_i \in \mathbb{Z}_p^{(2N+k)\times 2N}$ are consecutive rows of B formed as

$$C_i = \begin{pmatrix} B_i \\ B_{i+1} \\ B_{i+2}|_{[1:k,:]} \end{pmatrix}$$

for every $i = 1, \dots, n$ and an attack parameter $k \in \{1, \dots, N\}$. To ensure the well-definedness of the equation, set $B_{i+n} = B_i$. The matrix generation of C_i is displayed in Figure 3. Each matrix C_i will be utilized to test $i_0 = i$ for $i \in \{1, \dots, n\}$.

To do so, note that from (2), we directly obtain

$$C_i D = \begin{pmatrix} \epsilon_i \\ \epsilon_{i+1} \\ \epsilon_{i+2}|_{[1:k,:]} \end{pmatrix} = \overline{\epsilon_i} \tag{4}$$

for every $i = 1 \dots, n$. To facilitate the recovery of the index of interest, the authors formulated a CVP on the reduced-dimension lattices L_i. Specifically, they investigated

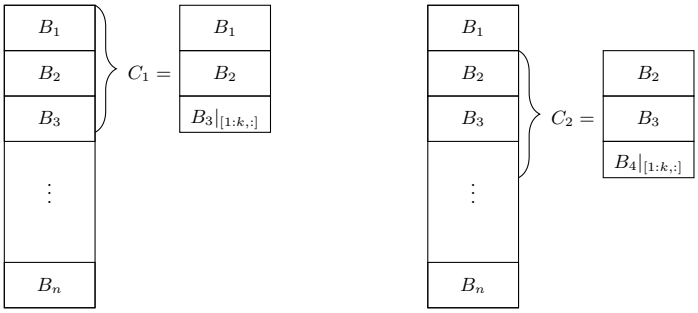

Figure 3: Generation of matrices C_i defining the lattice L_i in (3) within the attack of Liu and Bi [LB16] shown for $i = 1$ (left) and $i = 2$ (right).

CVP instances with target vectors $t_j = (0, \ldots, 0, q, 0, \ldots, 0) \in \mathbb{Z}^{2N+k}$ for $j = 1, \ldots, N$, where the value q is located in the j-th position. The value q in the target vector aligns with the amplified diagonal entries of the noise matrix ϵ_{i_0}, enabling the adversary to distinguish the desired index i_0.

Based on the provided definition, the authors demonstrated [LB16, Theorem 2] that, when considering the lattice L_{i_0}, the closest vector to t_j is, apart from the sign, the j-th column of $\overline{\epsilon_{i_0}}$ as given in (4) with high probability. However, the columns of $\overline{\epsilon_{i_0}}$ have a specific structure: the elements are restricted to $\{-1, 1\}$, except for values corresponding to the diagonal entries of ϵ_{i_0}, which are $\pm q$. Due to this highly non-random structure, we can distinguish L_{i_0} from other lattices L_i, $i \neq i_0$, with high probability by analyzing the closest lattice vector.

To solve the CVP, the authors utilized Kannan's embedding technique [Kan87]. Initially, a short lattice basis S of L_i was constructed. Using this short basis, the embedded lattice

$$L_{i,j} = \left\{ y \in \mathbb{Z}^{2N+k+1} : y = B_{\text{emb}} x \text{ for some } x \in \mathbb{Z}^{2N+1} \right\}$$

with

$$B_{\text{emb}} = \begin{pmatrix} S & t_j \\ 0 & 1 \end{pmatrix}$$

is defined. As discussed in Section 2, solving the SVP on $L_{i,j}$ yields the solution to the CVP on L_i with respect to t_j if the parameters are appropriately chosen.

To summarize, the attack proceeds as follows. Initially, we iterate over all lattices L_i and test $i_0 = i$ for $i \in \{1, \ldots, n\}$ based on L_i. Therefore, consider the target vectors t_j for $j = 1, \ldots, N$. If the closest lattice vector to t_j consist with the structure of a column of $\overline{\epsilon_{i_0}}$, then we can conclude that $i_0 = i$. In all other cases, we proceed with the next lattice. To solve the CVP for the target vector t_j, transform it into an SVP on the embedded lattice $L_{i,j}$. If the solution exhibits the specified structural properties, we identify the current index i as the desired file index, thereby recovering the target file. Within this attack, the choice of k represents a trade-off between computational cost, which increases with the value of k, and the growing gap between the shortest and second shortest vector within the embedded lattice, which also increases with k. This larger gap facilitates the identification of the shortest vector.

Remark 1. Instead of restricting the non-zero entry in t_j to the first N out of the $2N + k$ positions, an alternative strategy could involve varying the position of q in t_j across all possible positions. This approach would entail examining a larger number of target vectors.

Meanwhile, we only need to study a fraction of all lattices L_i. While the overall number of SVPs to be solved is similar for both approaches, the complexity of this approach is slightly better. The reason behind this improvement is the ability to reuse the short lattice basis S of L_i for each target vector, thereby enhancing the computational efficiency of the algorithm.

Remark 2. It is important to note that, from a mathematical perspective, iterating over all target vectors t_j, $j = 1, \ldots, N$ is not required since solving the CVP exactly with the first target vector would already yield the desired result. Nevertheless, lattice reduction algorithms are inherently approximative in nature. By iterating through multiple target vectors t_j, the risk of obtaining a false result is mitigated, thereby enhancing the reliability of the algorithm.

In [LB16], the authors empirically validated the efficacy of their proposed attack through numerical experiments conducted on a database with $n = 9$ files. However, there is a linear dependency between the number of files in the database and the number of lattice reductions needed for the attack. More specifically, even if only one target vector t_j is considered for each lattice L_i as suggested in Remark 2, the average number of CVPs that need to be solved is $\frac{n}{2}$, with the worst-case scenario requiring n CVPs. Taking into account the large computational time for lattice reduction algorithms, this linear dependency makes the attack impractical for databases exceeding a certain size. In [LB16], the runtime of a single CVP instance utilizing the BKZ algorithm [SE94] for parameters analogous to those proposed in [AMG08] is stipulated to be approximately 3000 seconds. This results in substantial attack times, even for relatively small parameter sizes.

5 An improved attack

While the attack strategy outlined in the previous section is valid for small databases, its scalability is compromised by a time complexity that grows linearly with the number of files in the database. To address this issue, we introduce a novel two-stage approach that combines the former attack with a preprocessing step designed to decrease the number of blocks in the matrix $B = (B_1, \ldots, B_n)^T$. By iteratively eliminating blocks B_i that do not match the block B_{i_0}, we generate a condensed query matrix that can be efficiently processed using the former attack by Liu and Bi, thus extending its applicability to larger databases.

To obtain a logarithmic instead of a linear complexity in the number of files n, we proceed as in a binary search and initially split $B = (B_1, \ldots, B_n)^T$ into two parts

$$B|_{[1:3lN,:]} = \begin{pmatrix} B_1 \\ \vdots \\ B_{3l} \end{pmatrix} \text{ and } B|_{[3lN+1:nN,:]} = \begin{pmatrix} B_{3l+1} \\ \vdots \\ B_n \end{pmatrix},$$

where $l = \lceil \frac{n}{6} \rceil$. By selecting l in this manner, we ensure that each part contains approximately $\frac{n}{2}$ of the n matrices B_i. In the subsequent analysis, we first examine the matrix $B|_{[1:3lN,:]}$ to determine whether B_{i_0} is included within this part of B. If i_0 is found to be within the range $\{1, \ldots, 3l\}$, we proceed with the first matrix and disregard the second; otherwise, we move to the second matrix and disregard $B|_{[1:3lN,:]}$.

To determine whether i_0 is included in the first $3l$ indices, we further divide the matrix

$B|_{[1:3lN,:]}$ into blocks of three matrices each and sum those blocks up such that we obtain

$$H = \begin{pmatrix} H_1 \\ H_2 \\ H_3 \end{pmatrix} = \sum_{i=0}^{l-1} \begin{pmatrix} B_{1+3i} \\ B_{2+3i} \\ B_{3+3i} \end{pmatrix}. \tag{5}$$

Summing blocks reduces the number of candidate matrices while preserving the overall structure, which can be illustrated as follows. Adding two matrices B_i and B_j results in

$$B_i + B_j = [(P_i + P_j)M_1 \,|\, (P_i + P_j)M_2 + (\epsilon_i + \epsilon_j)]\Delta$$

for $i, j \in \{1, \ldots, n\}$. Given that $P_i + P_j$ is again a random matrix, the structure of the sum remains similar to the original, with the exception of the noise term. The noise matrix $\epsilon_i + \epsilon_j$ in the sum is now characterized by entries in $\{-2, 0, 2\}$, where each of ± 2 occurs with a probability of $\frac{1}{4}$ and 0 occurs with a probability of $\frac{1}{2}$. Generalizing this to the summation of l matrices yields

$$H_j = [\tilde{P}_j M_1 \,|\, \tilde{P}_j M_2 + \gamma_j]\Delta$$

for $j = 1, 2, 3$ with

$$\tilde{P}_j = \sum_{i=0}^{l-1} P_{j+3i}$$

being a random matrix. The noise matrix γ_j is given by

$$\gamma_j = \sum_{i=0}^{l-1} \epsilon_{j+3i}$$

for $j = 1, 2, 3$, such that the entries of γ_j follow a shifted binomial distribution either on $\{-l, \ldots, l\} \cap (2\mathbb{Z})$ if l is even or on $\{-l, \ldots, l\} \cap (2\mathbb{Z} + 1)$ if l is odd. We can directly transfer the bridging relation (2) to the summed matrices as

$$HD = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{pmatrix} = \gamma. \tag{6}$$

Study the lattice defined by the summed matrix H as

$$L_p(H) = \{y \in \mathbb{Z}^{3N} \colon y = Hx \mod p \text{ for some } x \in \mathbb{Z}_p^{2N}\}. \tag{7}$$

From the bridging relation (6), we directly obtain the following relation between γ and the lattice $L_p(H)$.

Lemma 1. *Let $L_p(H)$ denote the lattice defined by (7), where H is derived from the summation of submatrices of B as specified in (5). Then each column of $\gamma = (\gamma_1, \gamma_2, \gamma_3)^T$ represents a lattice vector within $L_p(H)$ having entries in a subset of $\{-l, \ldots, l\}$. Consequently, for small values of l, the columns are short lattice vectors in $L_p(H)$.*

In analogy to the former attack, our objective is to solve a CVP on the lattice $L_p(H)$. To this end, we define a set of target vectors $t_j = (0, \ldots, 0, q, 0, \ldots, 0)$ for $j = 1, \ldots, 3N$, where the value q is placed at the j-th position. It is worth noting that, as highlighted in Remark 1, our approach employs a larger set of target vectors, yet reduces the number of lattices utilized compared to the former attack. This modification is motivated by an improvement in computational efficiency.

The subsequent theorem establishes that, for sufficiently small values of l, it is possible to decide whether B_{i_0} is contained in H or not.

Theorem 1. *Suppose that the number of additions l is bounded from above such that*

$$N_{\sqrt{3lN}}(0) \leq p^{N-2},$$

where $N_R(x)$ is the number of integer points within the $3N$-dimensional sphere as defined in Section 2. Assume that the desired file index i_0 is an integer such that $i_0 \in \{1, \dots, 3l\}$. Then there is an $a \in \{1, 2, 3\}$ such that B_{i_0} is a summand in H_a. Let $L_p(H)$ be the lattices as defined in (7) and consider the target vector t_j, where the index j satisfies

$$(a-1)N < j \leq aN.$$

In other words, we are considering a target vector with a non-zero entry within the same block as B_{i_0}. Then

$$u = j - (a-1)N \in \{1, \dots, N\}$$

is the row within block a, in which t_j has the non-zero entry. If the corresponding diagonal entry in ϵ_{i_0} equals $\epsilon_{i_0}[u, u] = q$, then the probability that the closest lattice vector with respect to t_j is given by $\gamma_{[:,u]}$ can be approximated by

$$\exp\left(-p^{-2}\frac{p^{3N}}{p^{3N}-1}\right).$$

Conversely, if $\epsilon_{i_0}[u, u] = -q$ then with the same approximate probability, the closest lattice vector with respect to t_j is given by $-\gamma_{[:,u]}$.

Proof. Initially, let us examine the vector $\gamma_{[:,u]}$. As a result of the summation, for every $v \in \{1, \dots, 3N\} \setminus \{j\}$, the v-th entry of this vector satisfies

$$\gamma_{[v,u]} \in \{-l, \dots, l\}.$$

In contrast, when $v = j$, the corresponding entry is constrained to the range

$$\gamma_{[j,u]} \in \{-l+1, \dots, l-1\} \pm q$$

such that

$$\begin{aligned} || \pm \gamma_{[:,u]} - t_j|| &\leq \sqrt{(3N-1)l + (l-1)} \\ &\leq \sqrt{3Nl}. \end{aligned}$$

Note that, due to the entries of γ being distributed according to a shifted binomial distribution, the length of the columns will be even shorter with high probability.

As a subsequent step, we demonstrate that, with high probability, there exists no lattice vector with a distance to the target vector that is equal to or shorter than this threshold. To this end, let $R > 0$ be fixed and recall that $B_R(x)$ denotes the $3N$-dimensional sphere of radius R centered at $x \in \mathbb{R}^{3N}$. We establish a lower bound for the probability that no lattice vector has a distance to t_j that is less than or equal to R, which can be expressed as

$$\begin{aligned} &P(\nexists\, y \in L_p(H) : ||y - t_j|| \leq R) \\ =&P(\nexists\, y \in B_R(t_j) \cap L_p(H)) \\ =&P(\nexists\, y \in B_R(t_j) \cap \mathbb{Z}^{3N} \colon y = Hx \mod p \text{ for some } x \in \mathbb{Z}_p^{2N}), \end{aligned}$$

where we used the definition of the lattice $L_p(H)$ in (7). We stress that H is not uniformly random as a matrix in $\mathbb{Z}_p^{3N \times \S N}$, as it inherits structure from the PIR construction. However,

since $\Delta \in GL_N(\mathbb{Z}_p)$ is invertible, for every fixed nonzero vector $x \in \mathbb{Z}_p^{2N}$, we may write $\Delta x = (x_1, x_2)^T \in \mathbb{Z}_p^N \times \mathbb{Z}_p^N$. Thus, each block of Hx takes the form

$$\begin{aligned} H_j x &= \tilde{P}_j(M_1 x_1 + M_2 x_2) + \gamma_j x_2 \\ &= \tilde{P}_j v + \gamma_j x_2, \end{aligned}$$

for $v = M_1 x_1 + M_2 x_2$. While the matrices M_1 and M_2 are shared across all blocks, the matrices $\tilde{P}_j$ and γ_j differ between blocks and arise from independent randomness. In particular, the term $\tilde{P}_j v$ is close to uniformly distributed over $\mathbb{Z}_p^N$ since $\tilde{P}_j$ is the sum of independent random invertible matrices, and the additive term $\gamma_j x_2$ acts as an additional random shift. Hence, the vector $Hx \mod p$ can be regarded as approximately uniformly distributed in $\mathbb{Z}_p^{3N}$. Consequently,

$$\begin{aligned} P(\nexists\, y \in L_p(H) : ||y - t_j|| \leq R) &\approx \prod_{y \in B_R(t_j) \cap \mathbb{Z}^{3N}} P(\nexists\, x \in \mathbb{Z}_p^{2N} : y = Hx \mod p) \\ &\approx \prod_{y \in B_R(t_j) \cap \mathbb{Z}^{3N}} \prod_{x \in \mathbb{Z}_p^{2N}} (1 - P(y = Hx \mod p)), \end{aligned}$$

where

$$P(y = Hx \mod p) = p^{-3N}.$$

It follows that the probability of no lattice vector being within a distance of R or less from t_j can be approximated by

$$\begin{aligned} P(\nexists\, y \in L_p(H) : ||y - t_j|| \leq R) &\approx (1 - p^{-3N})^{p^{2N} N_R(t_j)} \\ &= \exp\left(p^{2N} N_R(t_j) \log(1 - p^{-3N})\right). \end{aligned}$$

Due to our assumption, for the radius $R = \sqrt{3Nl}$ we obtain

$$N_{\sqrt{3lN}}(t_j) \leq p^{N-2}.$$

For the overall probability that no lattice vector with distance shorter or equal to R exists, this yields

$$\begin{aligned} P(\nexists\, y \in L_p(H) : ||y - t_j|| \leq \sqrt{3Nl}) &\approx \exp\left(p^{-2+3N} \log(1 - p^{-3N})\right) \\ &= \exp\left(-p^{-2+3N} \sum_{k=1}^{\infty} \frac{p^{-3Nk}}{k}\right) \end{aligned}$$

using the series representation of $\log(1 - x)$. Finally,

$$\begin{aligned} P(\nexists\, y \in L_p(H) : ||y - t_j|| \leq \sqrt{3Nl}) &\approx \exp\left(-p^{-2+3N} \sum_{k=1}^{\infty} p^{-3Nk}\right) \\ &= \exp\left(-p^{-2} \frac{p^{3N}}{p^{3N} - 1}\right), \end{aligned}$$

which finishes the proof. □

Remark 3. The formulation of Theorem 1 is quite technical, such that we want to give a more accessible interpretation. From an intuitive perspective, it is evident that the number of additions l must be bounded from above. If l would be unbounded, the difference between ϵ_{i_0} with $\pm q$ on its diagonal and all other ϵ_i becomes negligible in the summed

matrix H. The bound given in the theorem is implicit, requiring that l is chosen such that the number of integer points inside a $3N$-dimensional sphere of radius $\sqrt{3Nl}$ centered at the origin is at most p^{N-2}. Utilizing the approximation on $N_R(x)$ from Section 2, this implicit definition can be translated into an approximate bound on l. In detail, we obtain the upper bound l_{max} of l as

$$\begin{aligned} l_{max} &\approx \frac{\Gamma\left(\frac{3N}{2}+1\right)^{\frac{2}{3N}}}{3N\pi} p^{\frac{2}{3}(1-\frac{2}{N})} \\ &\approx \frac{1}{2\pi e} p^{\frac{2}{3}(1-\frac{2}{N})}, \end{aligned}$$

using that

$$\lim_{x\to\infty} \frac{\sqrt[x]{\Gamma(x+1)}}{x} \to e^{-1}.$$

The initial paper proposed values of $p = 2^{60} + 325$ and $N = 50$, yielding an estimated maximum length $l_{max} \approx 2.12 \cdot 10^{10}$. Notably, our approach involves adding at most $\lceil \frac{n}{6} \rceil$ blocks. Given the constraint on database size, where $n \leq 10,000$, it is evident that in this case l significantly falls below l_{max}. Consequently, for practical applications and parameters as suggested in [AMG08], the assumption on l can be readily fulfilled.
Consider a matrix H that incorporates B_{i_0}. According to Theorem 1, for a suitably chosen target vector t_j, the closest vector to t_j is given by a particular column of γ with probability approximately

$$\exp\left(-p^{-2}\frac{p^{3N}}{p^{3N}-1}\right) \approx 1 - p^{-2}.$$

Substituting the proposed value $p = 2^{60} + 325$ into this expression yields an approximate probability of $(1 - 7.52 \cdot 10^{-37})$ that the closest vector is indeed given by this column of γ. Although the probability obtained is already remarkably high, it is reasonable to expect an even higher probability of success in practical applications. This is because the proof is based on a worst-case analysis, considering the maximum possible value of l and the maximum possible length of the column of γ. In reality, the actual values of these parameters are likely to be more favorable, leading to an even higher probability of success. Consequently, it is highly unlikely that this theorem will ever fail in practical applications.

Under the assumptions of Theorem 1, the closest lattice vector with respect to t_j exhibits a distinctive pattern with high probability. Specifically, dependent on l, the entries of this vector are either all even or all odd, with the notable exception of the j-th entry, which possesses the opposite parity. Remarkably, the occurrence of such a configuration is exceedingly rare in random instances, such that this behavior of the closest lattice vector can serve as the foundation for our attack.

As in [LB16], we use Kannan's embedding technique to solve the CVP on $L_p(H)$. Therefore, let $S \in \mathbb{Z}^{3N\times 3N}$ be a short basis of $L_p(H)$ and let the embedded lattice be given by

$$L_{H,j} = \left\{ y \in \mathbb{Z}^{3N+1} : y = B_{\text{emb}} x \text{ for some } x \in \mathbb{Z}^{3N+1} \right\}$$

with

$$B_{\text{emb}} = \begin{pmatrix} S & t_j \\ 0 & M \end{pmatrix}.$$

By Theorem 1, if B_{i_0} is contained in H and the index of the target vector is appropriately selected, the shortest vector is effectively determined by a column of γ. Building

upon this result, our proposed attack proceeds as follows. Initially, we partition the matrix B into two segments and compute the sum of the first segment to obtain the matrix H. We then define the lattice $L_p(H)$ and calculate a short basis S of $L_p(H)$. To mitigate computational complexity, as argued in Remark 2, we restrict our attack to the target vectors t_1, t_{N+1}, and t_{2N+1}. For these three target vectors, we solve the SVP on the embedded lattice. If the solution exhibits a structural consistency with a column of γ, we can infer that B_{i_0} is contained in H. Furthermore, by retaining the target vector that yields the successful outcome, we can even identify the specific block of H in which B_{i_0} is contained. This enables us to not only disregard the second half of the matrix B but also eliminate the two blocks of H that do not contain B_{i_0}, thereby further reducing the number of blocks to be considered in the subsequent iteration to l. Conversely, if B_{i_0} is not present in H, we can deduce that it is contained in the second half of B, and the procedure continues with this half. In average, this reduces the number of matrices to be considered by a factor of $\frac{1}{3}$. By recursively applying this approach, we can further narrow down the number of matrices until it reaches a manageable size for the attack by Liu and Bi. Denote with $s \in \mathbb{N}$ the threshold value, which triggers the former attack within our two-step process. The complete attack, including the novel preprocessing step as well as the original attack by Liu and Bi applied to a subset of blocks, is presented in Algorithm 1. Note that the only assumption required for the attack to succeed is the restriction on the database size given in Theorem 1, which is a mild assumption as argued in Remark 3. If, however, the database size exceeds this size, the attack can instead be applied to parts of the query separately. While this increases the computational cost, the attack remains feasible for arbitrary database sizes.

Although the PIR scheme of Aguilar Melchor and Gaborit [AMG08] considered in this work is neither the most recent nor the most efficient construction, it represents one of the earliest lattice-based PIR schemes and continues to serve as a point of reference for subsequent research. In particular, it provides a clean and accessible setting to demonstrate the potential of our binary-search-like technique. Importantly, this technique does not depend on the specific initialization of the PIR scheme, but only on the availability of functionality that reveals whether a queried index belongs to a given subset of indices. Consequently, similar binary-search-like approaches may be applicable to a broader class of PIR constructions exhibiting comparable structural properties. We emphasize that the binary-search-like technique is not limited to lattice-based constructions and can be applied to schemes relying on other hardness assumptions.

As a first illustrative example, consider the code-based PIR scheme of [HHWZ20]. This scheme was previously broken in [BL21] by exploiting the fact that certain submatrices of the query matrix reveal the queried index through rank computations. In that attack, identifying the desired index in a database of size n requires, on average, $\frac{n}{2}$ rank evaluations. In contrast, by applying our binary-search-like technique, the number of required rank computations can be reduced to approximately $\log(n)$. We note, however, that detecting rank deficiencies necessitates retaining a sufficient number of rows, implying a lower bound on the database size and the block size for the attack to succeed. Nevertheless, since PIR protocols typically assume large databases, this requirement is not restrictive in practice. While the attack in [BL21] already compromises the scheme, our method significantly improves its efficiency.

Our binary-search-like idea also inspired the attack in [LB25] on the code-based PIR scheme by Hollanti and Verna [HV25]. The PIR scheme in [HV25] builds on the earlier construction in [HHWZ20] but especially aims to mitigate the subquery attack in [BL21] by spreading the noise over all blocks of the query matrix. As a trade-off, the user must send two query matrices instead of one to query one file. The attack in [LB25] proceeds in two stages. In the first step, the adversary constructs an auxiliary matrix that reveals a

set of candidate relations between the noise of two fixed blocks of the first query matrix. Intuitively, this step does not yet determine the correct relation, but narrows down the possibilities. In the second step, each candidate relation is tested individually to identify the correct one. Once this relation is known, the attacker can use the second query matrix to check whether one of the blocks corresponds to the desired database index. Again, the attack requires repeated rank computations over a large field $\mathbb{F}_q$. A key observation in Section 4 of [LB25] is that the number of required tests within the second step can be reduced using our binary-search-like technique. Instead of checking all $(\delta - 1)$ candidate relations sequentially, one can identify the correct relation in $\log(\delta)$ steps, where δ depends on the file size of the database entries. In contrast to the example above, this optimization applies specifically to the selection of the correct relation from the candidate set and does not reduce the complexity of other parts of the attack. Nevertheless, since PIR schemes typically involve large file sizes, this yields a significant improvement in overall runtime. Moreover, the fact that the binary-search-like idea applies only to a specific substep of the attack highlights its flexibility, as it can be incorporated into individual components of more complex procedures.
In principle, our binary-search-like technique is not restricted to PIR schemes and could be applied to attacks on arbitrary cryptographic primitives, regardless of the underlying hardness assumption. However, this would require that some part of the ciphertext or differences between multiple ciphertexts are distinguishable in a way that leaks information about the underlying plaintexts. While PIR schemes often exhibit such distinguishability more naturally (e.g. though encryptions of vectors containing only zeros except for a single one), it is unlikely that other cryptographic primitives display similar behavior. Therefore, we expect the most natural applications of our technique within PIR settings or related primitives.

5.1 Computational Costs

To assess the practicability of our attack, we now analyze its computational cost. In each iteration of the preprocessing phase, we construct the matrix H by summing l matrices of size $3N \times 2N$. Since each matrix addition requires $O(N^2)$ operations over $\mathbb{Z}_p$, this step incurs a total cost of $O(lN^2)$. Subsequently, we solve a CVP on the lattice $L_p(H) \subset \mathbb{Z}_p^{3N}$. Using Kannan's embedding technique, this is reduced to a SVP in dimension $d = 3N + 1$. Denoting the cost of the SVP solver in dimension d with $\mathcal{L}(d)$, a single preprocessing iteration therefore requires

$$O(lN^2 + \mathcal{L}(d))$$

operations over $\mathbb{Z}_p$. As the number of blocks l decreases geometrically throughout the preprocessing phase, the total cost of all summations is bounded by $O(nN^2)$. Moreover, the number of CVP instances solved during preprocessing is $O(\log(n))$ due to our binary-search like approach. Hence, the overall cost of the preprocessing phase is

$$O(nN^2 + \log(n)\mathcal{L}(3N + 1)).$$

To make this estimate more explicit, we consider the cost of a particular SVP solver. For instance, when using the LLL algorithm, solving a SVP instance via Kannan's embedding technique requires $\mathcal{L}(d) = O(d^4 \log^2(p))$ [NS16, Theorem 2] operations, which yields $\mathcal{L}(3N + 1) = O(N^4 \log^2(p))$. Hence, for the parameter regime considered in this work, the lattice reduction term dominates the matrix operations and the overall cost of the preprocessing is given by

$$O(\log(n)\mathcal{L}(3N + 1)).$$

Algorithm 1: Improved attack on the lattice-based PIR scheme

Data: Query matrix $B = (B_1, \ldots, B_n)$, threshold s, attack parameter k
Result: Index of interest i_0

Function `Main`(B, s, k)**:**
  Blocks= $[B_1, \ldots, B_n]$
  RemainingBlocks, candidates = Preprocessing(Blocks, s)
  i_0 = LiuBiAttack(RemainingBlocks, candidates, k)
  return i_0

Function `Preprocessing`(*Blocks,s*)**:**
  stepsize $sz = 3$
  candidates = [1,. . . ,n]
  while $|Blocks| \geq s$ **do**
    $l = \lceil \frac{|\text{Blocks}|}{6} \rceil$
    $\text{Part}_1 = \text{Blocks}[1 : 3l]$
    $H = \sum_{i=0}^{l-1} \begin{pmatrix} Blocks[1+3i] \\ Blocks[2+3i] \\ Blocks[3+3i] \end{pmatrix}$
    define lattice $L_p(H)$
    found = FALSE
    for $i \in \{1, 2, 3\}$ **do**
      target vector $\tilde{t}_i = q \cdot e_{(i-1)N+1}$
      find closest vector to $\tilde{t}_i$ in $L_p(H)$
      if *closest vector is valid* **then**
        found=TRUE
        $\text{Blocks} = \text{Part}_1[i : sz : 3l]$
        $\text{candidates} = \text{candidates}[i : sz : 3l]$
        break
    if *found=FALSE* **then**
      $\text{Blocks} = \text{Blocks}[3l + 1 : end]$
      $\text{candidates} = \text{candidates}[3l + 1 : end]$
  return Blocks, candidates

Function `LiuBiAttack`(*Blocks, candidates, k*)**:**
  $n' = |\text{candidates}|$
  for $i = 1, \ldots, n'$ **do**
    $C_i = \begin{pmatrix} Blocks[i] \\ Blocks[i+1] \\ Blocks[i+2]|_{[1:k,:]} \end{pmatrix}$
    define lattice L_i as in (3)
    target vector $t = qe_1$
    find closest vector to t in L_i
    if *(closest vector* $- t$*) in* $\{-1, 1\}^{2N+k}$ **then**
      return candidates[i]

Note that this holds for other SVP solvers likewise, since they typically come at even higher computational costs. We note that the preprocessing phase can be further optimized by caching partial sums of the matrices B_i. Since the matrices H in different iterations are constructed from overlapping subsets of the B_i, intermediate results can be reused to

reduce the number of matrix additions. This leads to a practical reduction in runtime at the expense of increased memory usage, while the asymptotic complexity remains unchanged. After preprocessing, the attack of Liu and Bi is applied to the reaming $\tilde{s} \leq s$ blocks. This requires solving at most $\tilde{s}$ additional CVP instances, each with cost $O(\mathcal{L}(2N + k + 1))$. Hence, the total runtime of the attack is given by

$$O((\log(n) + s)\mathcal{L}(3N + 1)),$$

where we used that $3N + 1 \geq 2N + k + 1$. We summarize this result in the following Lemma.

Lemma 2. *The algorithm outlined above recovers the desired file index i_0 with a runtime complexity of $O((\log(n) + s)\mathcal{L}(3N + 1))$, whereas the attack in [LB16] has a runtime complexity of $O(n\mathcal{L}(2N + k + 1))$.*

Remark 4. For large values of N, a further optimization can be applied by truncating the lower matrix H_3 of H in (5) to have only k rows, where k is an attack parameter ranging from 1 to N. This reduces the complexity of the new attack to $O((\log(n)+s)\mathcal{L}(2N+k+1))$, thereby providing an additional performance improvement.

The attack by Liu and Bi becomes impractical for large databases because the number of CVPs to solve scales linearly with n. Specifically, in the worst-case scenario, their attack requires solving n CVPs, while on average, it necessitates solving $\frac{n}{2}$ CVPs. Our present approach exhibits a logarithmic dependence on n, making the attack suitable for all database sizes. To underscore the low complexity of our attack, we examine both the worst-case and average-case scenario. Notably, the worst-case scenario corresponds to the instance where $i_0 = n$, necessitating the solution of three CVPs in each iteration without yielding a successful outcome, thereby proceeding with the latter half of the matrix.

A theoretical approximation of the number of CVPs to be solved is given by

$$\#\text{CVPs}_{\text{WorstCase}} \approx 3\frac{\log(n) - \log(t)}{\log(2)} + 0.5s. \tag{8}$$

In contrast, for the average case, we derive

$$\#\text{CVPs}_{\text{AverageCase}} \approx 2.5\frac{\log(n) - \log(t)}{\log(3)} + 0.25s. \tag{9}$$

Note that the two formulas first account for the number of CVPs in each iteration during preprocessing, multiplying that by the expected number of iterations. If the threshold s of remaining blocks is surpassed, the attack by Liu and Bi is initiated with approximately $0.5s$ blocks remaining. In the worst-case scenario, the entire number of remaining blocks must be processed; however, on average, the attack will require searching only half of the remaining blocks. To illustrate the practical implications for the complexity, we consider a threshold value of $s = 6$ and plotted the exact as well as the approximate number of CVPs to be solved in our attack for various database sizes n in Figure 4. The variability within the exact worst-case curve can be attributed to the interplay between the novel precomputing stage and the attack in [LB16]. As the size of n fluctuates, the attack in [LB16] is initiated with matrices of varying dimensions, resulting in a non-monotonic curve. In contrast, the averaging process in the average case largely mitigates the impact of disparate starting lengths for the former attack, thereby smoothing the curve. The approximate formulas presented in (8) and (9) demonstrate a satisfactory level of precision in approximating the actual number of CVPs for both scenarios.

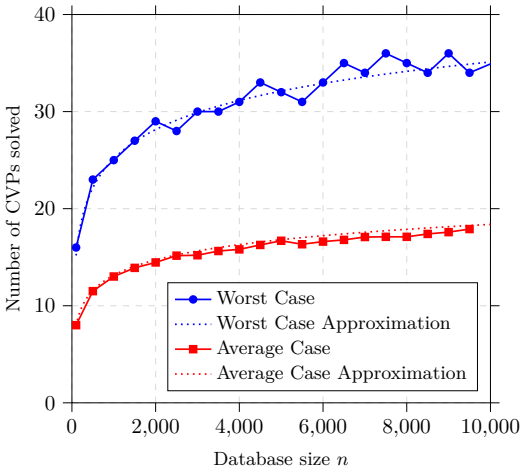

Figure 4: Exact and approximate number of CVPs to be solved in our attack in the worst case and the average case for different database sizes n and a threshold value of $s = 6$.

6 Numerical results

Using SageMath [The24], we implemented our attack following the pseudocode given in Algorithm 1. As outlined in Section 5.1, we store particular submatrices, which are needed for multiple iterations of H within the precomputing step, to avoid unnecessary computational costs. Following [AMG08], we first set $l_0 = 20$, $q = 2^{2l_0-1}$, $N = 50$, and $p = 2^{60} + 325$. The suggested database size for these parameters is $n \leq 10,000$. We continued the preprocessing until $s = 6$ or fewer blocks were left. Subsequently, we initiated the attack by Liu and Bi [LB16] using $k = 34$ as recommended. For all lattice reductions, we employed the LLL algorithm. For the embedded lattice, we set $M = 1$ as the embedding factor. This choice ensures that the gap between the shortest and the second shortest lattice vector is large. Meanwhile, it also increases the probability of false results. However, for this particular case, the procedure operates correctly, leading us to retain this parameter choice. The results are presented in Table 1. We display the minimal and maximal run time of the attack as well as the success rate, which is measured across 100 trials. All calculations were conducted on a laptop equipped with an Intel Core i7-1370P processor with 32 GB RAM. Regardless of the database size, the index of interest can be recovered reliably within minutes, demonstrating the complete breakdown of the scheme. For an easier comparison, the computational process is not executed in parallel. However, the described approach is particularly well-suited for parallel computing. By dividing the query matrix into separate blocks and searching within these blocks in parallel, the computational time can be further reduced.

Remark 5. It is worth noting that, for the specific parameters considered, the application of the LLL algorithm yields the index of interest with reasonable accuracy. In this particular case, solving a single CVP instance is relatively inexpensive compared to more advanced algorithms such as BKZ. However, our objective was to develop an attack that is viable for all possible parameter choices. In such a general cases, it is necessary to employ a more reliable lattice reduction technique, such as BKZ, which can provide more accurate results. Nevertheless, this comes at the cost of increased runtime with respect to the chosen blocksize. Consequently, the difference in performance between the former attack and our new attack becomes even more pronounced in the general case. For example, if we follow the assumptions of [LB16] with a single CVP instance requiring 3000s, finding the

index of interest in our attack scenario for a database of $10,000$ files requires 16.7 hours while the attack by Liu and Bi would need 174.6 days to finish in the average case.

Table 1: Minimal and maximal attack time as well as success rate over 100 trials for our improved attack with different database sizes n using $l_0 = 20$, $q = 2^{39}$, $N = 50$ and $p = 2^{60} + 325$. The attack by Liu and Bi [LB16] is triggered at $s = 6$ with $k = 34$.

n	Min. attack time (min)	Max. attack time (min)	Success rate (%)
100	0.82	1.56	100
1,000	1.10	1.58	100
5,000	1.29	2.76	100
10,000	1.45	2.94	100

To demonstrate the overall efficiency of our attack in comparison with the former attack proposed by Liu and Bi, we evaluate two different parameter sets as summarized in Table 2. Note that the parameter N determines the dimension of the lattice. Consequently, increasing N leads a higher security level. The parameters l_0, q and p are selected in accordance with [AMG08], namely $l_0 = \lceil \log_2(Nn) \rceil + 2$, $q = 2^{2l_0-1}$, and p as a prime satisfying $p > 2^{3l_0}$, where we set a maximal database size of $n = 10,000$ files. Furthermore, we chose the attack parameter k with $1 \leq k < N$. To assess performance across varying database sizes, we compare the runtime of our attack with that of the Liu-Bi approach. For consistency, both attacks employ the LLL algorithm to solve the SVP. In our attack, the preprocessing step reduces the number of blocks until $s = 6$ or less blocks remain. Note that in our attack, the runtime variance is small since the number of CVP instances to be solved differs only slightly between the worst- and the best-case scenario. In contrast, the runtime of the attack by Liu and Bi exhibits significant variability, ranging from the worst case of n CVP instances to the best case of a single instance. On average, however, approximately $\frac{n}{2}$ CVP instances must be solved to identify the desired index. Figure 5 and 6 illustrate the average runtime of both approaches for the two parameter sets listed in Table 2. The left-hand side presents results for smaller databases, while the right-hand side shows runtimes for larger databases on a logarithmic scale. For our attack, the reported values represent averages over 50 random trials. In the case of the attack of Liu and Bi, the index $i_0 = \frac{n}{2}$ is used to calculate the average runtime. All plotted points are obtained through numerical evaluation, while the dotted lines in the right-hand plots of Figure 5 and Figure 6 correspond to the best linear fit derived from the data shown on the left-hand side.

Table 2: Further parameter sets.

Parameter set	N	l_0	q	p	k
1	75	22	2^{43}	$2^{66} + 9$	40
2	100	22	2^{43}	$2^{66} + 9$	55

The results strongly support our theoretical analysis. While the runtime of the former attack by Liu and Bi grows linearly with n, our approach exhibits only logarithmic scaling. This demonstrates that the former becomes impractical beyond a certain database size, whereas our method remains efficient. Although solving single CVP instances takes significant time when increasing the security parameter N further or when changing to a parameter regime where an attacker would need to employ more complex solvers instead of LLL, our numerical analysis clearly implies the scalability of our attack and hence the

insecurity of the scheme. As a consequence, the scheme can be considered broken for all parameters and database sizes.

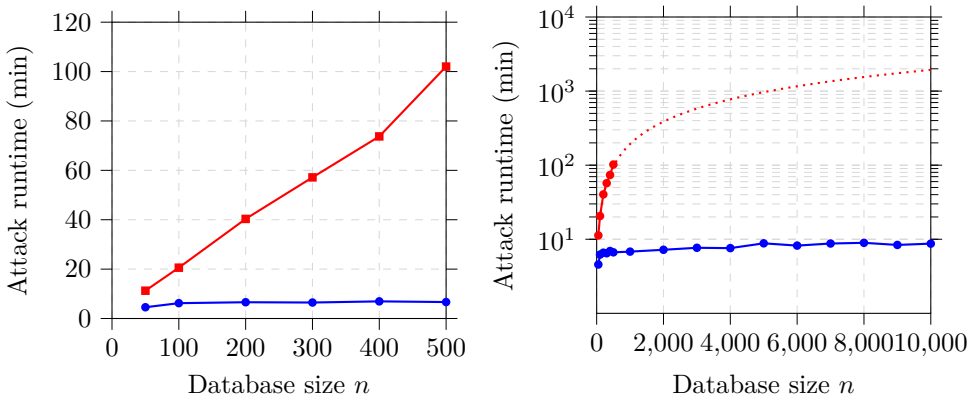

Figure 5: Average attacking time (in minutes) for the attack by Liu and Bi (red) and our proposed attack (blue). Results are shown for parameter set 1 (cf. Table 2) on a linear scale for small databases (left) and on a logarithmic scale for larger databases (right). The dotted red curve in the right-hand plot represents the best linear fit to the corresponding data points.

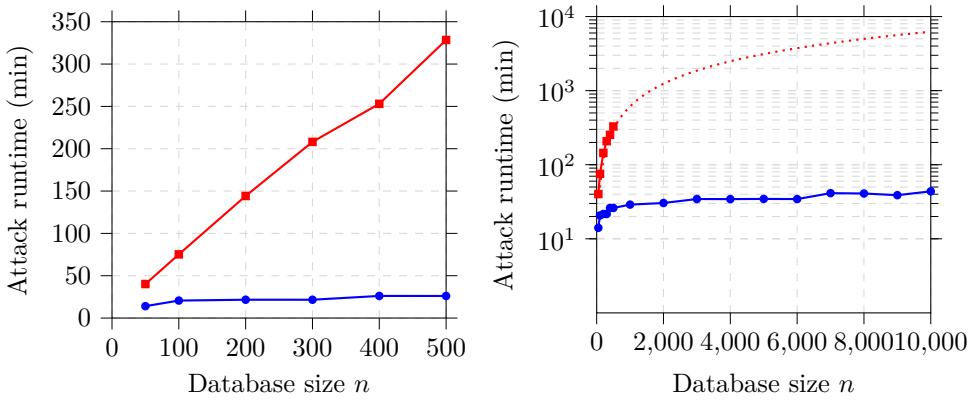

Figure 6: Average attacking time (in minutes) for the attack by Liu and Bi (red) and our proposed attack (blue). Results are shown for parameter set 2 (cf. Table 2) on a linear scale for small databases (left) and on a logarithmic scale for larger databases (right). The dotted red curve in the right-hand plot represents the best linear fit to the corresponding data points.

7 Conclusion

In this paper, we demonstrate a method for compromising the lattice-based PIR scheme introduced by Aguilar Melchor and Gaborit in [AMG08]. Although more efficient PIR schemes have been developed since, the original scheme remains an influential blueprint for many recent designs. Consequently, demonstrating its vulnerability to real-time attacks provides valuable insights that can inform future considerations and security assessments. Our attack on the scheme builds upon the approach proposed by Liu and Bi [LB16], which we augment with a novel preprocessing step. This enhancement significantly expands

the attack's applicability, making it effective against databases of arbitrary size, rather than just small ones. By utilizing a binary-search-like technique to presort the matrices, we dramatically reduce the number of lattice problems that need to be solved. This optimization yields a substantial decrease in complexity, from linear to logarithmic in the number of files within the database, thereby rendering the scheme insecure for all parameter choices. Furthermore, our numerical evaluations demonstrate the feasibility and reliability of our approach, with successful attacks executed within minutes on standard hardware. The binary-search-like preprocessing technique can be extended to other cryptographic schemes, provided that the ciphertext can be partitioned into chunks and there exists a function that evaluates a specific desired property on those chunks. This generalization underscores the result's importance beyond the single application outlined here. In particular, it suggests that future protocol designs should explicitly account for and mitigate such forms of distinguishability to prevent similar attacks. A natural direction for future work is to systematically analyze modern PIR constructions for such leakage. Furthermore, designing a code-based PIR scheme that avoids these structural weaknesses would be of particular interest, as it would increase diversity in the hardness assumptions underlying PIR.

Acknowledgments This work has been supported by funding from Agentur für Innovation in der Cybersicherheit GmbH. We thank the anonymous reviewers for valuable remarks and questions which helped improve the paper. Especially, we thank one of the reviewers for suggesting the application of our binary-search-like technique to the attack described in [BL21] as outlined in Section 5.

References

[Ajt98] Miklós Ajtai. The shortest vector problem in L2 is NP-hard for randomized reductions (extended abstract). In *30th Annual ACM Symposium on Theory of Computing*, pages 10–19, Dallas, TX, USA, May, 23-26, 1998. ACM Press. doi:10.1145/276698.276705.

[AMBFK16] Carlos Aguilar Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. XPIR : Private Information Retrieval for Everyone. *Proceedings on Privacy Enhancing Technologies*, avril 2016:155–174, April 2016. doi:10.1515/popets-2016-0010.

[AMG08] Carlos Aguilar Melchor and Philippe Gaborit. A fast private information retrieval protocol. In *2008 IEEE International Symposium on Information Theory*, pages 1848–1852, 2008. doi:10.1109/ISIT.2008.4595308.

[BL21] Sarah Bordage and Julien Lavauzelle. On the privacy of a code-based single-server computational PIR scheme. In *Cryptography and Communications*, volume 13, pages 519–526, 2021. doi:10.1007/s12095-021-00477-z.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 97–106, Palm Springs, CA, USA, Oct, 22-25 2011. IEEE Computer Society Press. doi:10.1109/FOCS.2011.12.

[CI95] Fernando Chamizo and Henryk Iwaniec. On the sphere problem. *Revista Matemática Iberoamericana*, 11(2):417–429, 1995. URL: http://eudml.org/doc/39471.

[CKGS98] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *Journal of the ACM*, 45(6):965–981, November 1998. doi:10.1145/293347.293350.

[CMS99] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT 99*, volume 1592 of *Lecture Notes in Computer Science*, pages 402–414, Prague, Czech Republic, May, 2-6 1999. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-48910-X_28.

[HHC+23] Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. One server for the price of two: Simple and fast single-server private information retrieval. In Joseph A. Calandrino and Carmela Troncoso, editors, *USENIX Security 2023: 32nd USENIX Security Symposium*, pages 3889–3905, Anaheim, CA, USA, Aug, 9-11 2023. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity23/presentation/henzinger.

[HHWZ20] Lukas Holzbaur, Camilla Hollanti, and Antonia Wachter-Zeh. Computational code-based single-server private information retrieval. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 1065—-1070, Los Angeles, CA, USA, 2020. IEEE Press. doi:10.1109/ISIT44484.2020.9174138.

[HV25] Camilla Hollanti and Neehar Vermal. CB-cPIR: Code-based computational private information retrieval, 2025. arXiv:2505.03407.

[Kan87] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12(3):415–440, 1987. URL: http://www.jstor.org/stable/3689974.

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *38th Annual Symposium on Foundations of Computer Science*, pages 364–373, Miami Beach, Florida, Oct, 19-22 1997. IEEE Computer Society Press. doi:10.1109/SFCS.1997.646125.

[LB16] Jiayang Liu and Jingguo Bi. Cryptanalysis of a fast private information retrieval protocol. In *Proceedings of the 3rd ACM International Workshop on ASIA Public-Key Cryptography*, AsiaPKC '16, pages 56–60, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2898420.2898427.

[LB25] Svenja Lage and Hannes Bartz. On the security of a code-based PIR scheme, 2025. arXiv:2507.19295.

[LLL82] Arjen K. Lenstra, Hendrik W. Lenstra, and Laszlo Lovasz. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982. doi:10.1007/BF01457454.

[MG02] Daniele Micciancio and Shafi Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Springer International Series in Engineering and Computer Science*. Springer Science & Business Media, New York, 2002. doi:10.1007/978-1-4615-0897-7.

[MO90] James E. Mazo and Andrew M. Odlyzko. Lattice points in high-dimensional spheres. *Monatshefte für Mathematik*, 110(1):47–61, 1990. doi:10.1007/BF01571276.

[NS16] Arnold Neumaier and Damien Stehlé. Faster LLL-type reduction of lattice bases. In *Proceedings of the 2016 ACM International Symposium on Symbolic and Algebraic Computation*, ISSAC '16, pages 373–380, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2930889.2930917.

[SC07] Radu Sion and Bogdan Carbunar. On the practicality of private information retrieval. In *ISOC Network and Distributed System Security Symposium – NDSS 2007*, San Diego, CA, USA, Feb. 28. - Mar. 2. 2007. The Internet Society. URL: https://www.ndss-symposium.org/ndss2007/practicality-private-information-retrieval/.

[SE94] Claus P. Schnorr and Martin Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming*, 66:181–199, 1994. doi:10.1007/BF01581144.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997. doi:10.1137/S0097539795293172.

[The24] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 10.3)*, 2024. URL: https://www.sagemath.org.

[vEB81] Peter van Emde-Boas. Another NP-complete partition problem and the complexity of computing short vectors in a lattice. Technical report, University of Amsterdam, Department of Mathematics, 1981.

[WAT18] Yuntao Wang, Yoshinori Aono, and Tsuyoshi Takagi. An experimental study of kannan's embedding technique for the search lwe problem. In Sihan Qing, Chris Mitchell, Liqun Chen, and Dongmei Liu, editors, *Information and Communications Security*, pages 541–553, Cham, 2018. Springer International Publishing. doi:10.1007/978-3-319-89500-0_47.