Verifiable Single-Server Private Information Retrieval

Xingfeng Wang[1] and Liang Zhao[1,2](✉)

[1] College of Cybersecurity, Sichuan University, Chengdu, China
xingfengW@yeah.net, zhaoliangjapan@scu.edu.cn
[2] HIFIVE Lennon Laboratory, Chengdu HiFive Technology Co., Ltd., Chengdu, China

Abstract. Single-server Private Information Retrieval (SPIR) allows a client to privately retrieve some data from a database stored on a server. While many SPIR schemes exist, these previous SPIR schemes are generally under the honest-but-curious server model. This model however is not suitable for many real world scenarios such as involving the untrusted cloud server. In this paper, we first propose an SPIR scheme that is based on the learning with (binary) errors assumption under the honest-but-curious server model. Specifically, compared with some previous SPIR schemes, our proposal provides a low communication complexity. Then, according to the above warm-up scheme, we introduce a Verifiable SPIR (VSPIR) scheme under the malicious server model where the server may provide some fraudulent answers. To the best of our knowledge, our scheme is the first practical VSPIR scheme that employs the probabilistic verification process. Finally, for our proposal, we present the theoretical analyses of the properties (i.e., correctness, privacy and security), and give the detailed implementation results.

Keywords: Learning with errors
Single-server private information retrieval
Probabilistic verification process

1 Introduction

1.1 Background

In the age of Internet accessing remote database is common and information is the most sought after and costliest commodity. In such a situation it is very important not only to protect information but also to protect the identity of the information that a user is interested in [11]. Private Information Retrieval (PIR) schemes are cryptographic schemes that enable users to retrieve records from public databases while keeping private the identity of the retrieved records [2]. In PIR schemes, a client is allowed to retrieve an entry from a server in possession of a database without revealing which entry is retrieved. The concept of PIR was first proposed in 1995 by Chor et al. [4]. There is a trivial solution consisting in sending the entire database regardless of the query. This solution has a high communication complexity of the database's size tb(at least $\log tb$ bits). Later, some schemes [3,16,18]that send less data have been proposed. Specifically, the Fully Homomorphic Encryption (FHE) and even the SomeWhat Homomorphic Encryption (SWHE) proposed by Gentry is known to imply the PIR scheme [3].

Moreover, in some practical scenarios, the server may provide the incorrect answers due to malicious behaviors or accidental failures. These scenarios can be defined as the malicious server model. Under this model, a PIR scheme can work effectively if the client should be able to identify the incorrect answers with overwhelming probability. This implies that how to verify the returned answers is a significant problem for a PIR scheme. Actually, for the honest-but-curious server model used in the previous work, it is assumed that the server is honest, which means that he follows the predefined scheme. From this point, this model is not very practical compared with the malicious server model. Then, constructing a PIR scheme that is secure in the malicious server model is well motivated and has been put forth by Beimel [2].

1.2 Related Work

In [18], Zhang and Safavi-Naini gave a verifiable multi-server PIR scheme where the servers may be malicious and provide some fraudulent answers. This scheme is an unconditionally t-private and computationally secure k-server verifiable PIR scheme in the honest-but-curious server setting. The drawback of this scheme is that it is too complicated to implement practically. Moreover, this PIR scheme does not work when all colluding servers host the database, which can be seen as the single malicious server setting..

In Sect. 5 of [3], the SWHE scheme is used to construct an asymptotically efficient Single-server PIR (SPIR) scheme based on the Learning With Errors (LWE) assumption. Specifically, this scheme employs some symmetric encryption scheme in the retrieval procedure. Using the most efficient symmetric scheme with the respect to the communication, the corresponding complexity of this scheme is $\mathcal{O}((\log n) + \kappa \mathsf{poly} \log(\kappa))$ (n is the database size and κ is the security parameter).

In [16], Vannet and Kunihiro proposed an SPIR scheme under the honest-but-curious server model relying on the unrelated Approximate GCD (AGCD) assumption. Assume the size of database is tb, which can be split into nb blocks of mw words of bb bits each, such that $nb \cdot mw \cdot bb = tb$. When nb cannot be decomposed in this way, pad the database with several bits. The database is denoted by a 2-dimensional array of words where each word is marked by two coordinates. Now use the set $\{b_{i,j} | 1 \le i \le nb, 1 \le j \le mw\}$ to denote the database, and write block u as $\{b_{u,j} | 1 \le j \le mw\}$. The security of this scheme is based on the AGCD assumption introduced in [6]. The assumption is said that given a random distribution of values $pq + \epsilon$ where $\epsilon \ll p$, the q has φ_q bits. Sample a set of this distribution, output p. In the single bit scheme, assume that the client wants to retrieve the block u consisting of $\{b_{u,j}\}_{1 \le j \le mw}$. The client samples a large random odd number p, and saves it as the secret key. He picks nb

random numbers q_i and ϵ_i, and computes $Q_i = pq_i + 2\epsilon_i + \delta_{i,u}$($\delta_{i,u}$ is the index vector where $\delta_{i,j} = 1$ if $i = j$, 0 otherwise). For each Q_i that the server received, compute $R_j = \sum_{i=1}^{nb} b_{i,j} Q_i$ and send it back to the client. On receiving R_j, the client decodes that $(R_j \mod p) \mod 2 = (\sum_{i=1}^{nb} b_{i,j}(pq_i + 2\epsilon_i + \delta_{i,u}) \mod q) \mod 2 = \sum_{i=1}^{nb} b_{i,j}(2\epsilon_i + \delta_{i,u}) \mod 2 = b_{u,j}$. In this scheme, p and q should be two large integers, which can guarantee that the scheme holds the security property. However, this scheme works under the honest-but-curious server model but not the malicious server model.

1.3 Open Problem

The previous work [9,16] related to SPIR is generally under the honest-but-curious server model. This model however is not suitable for many real-world scenarios such as involving the untrusted cloud server. From Zhang and Safavi-Naini's work [18], although a verifiable multi-server PIR scheme has been presented, constructing a Verifiable SPIR constructing a Verifiable SPIR (VSPIR) scheme under the malicious server model seems to be a difficult task. This is due to the fact that the protection of the input index depends on the heavy FHE scheme, which implies that the computational complexity is very high. To the best of our knowledge, there has not been a practical VSPIR scheme. Therefore how to construct a simple and pratical VSPIR is still an open problem.

1.4 Our Contributions

In this work, we present two main contributions. The first warmup one is to introduce an SPIR scheme based on the decision-LWE with binary error assumption under the honest-but-curious server model. Then, according to this scheme, we construct a VSPIR scheme under the malicious server model.

- **The SPIR scheme based on the decision-LWE assumption**. In our proposed SPIR, we use the database defined in [16]. We assume that a client wants to obtain the block u without revealing any information about u. The client uses a special variant of the encryption scheme with additive homomorphism in [7] to encrypt the query vector where the u-th elements is 1 and others are 0, then compute the query messages $\{Q_i\}$. A server computes $R_j = \sum_i^n b_{i,j} Q_i$ where R_i is equivalent to the encrypted $b_{u,j}$. Then the server sends R_j to the client. For each R_j, the client runs the homomorphic decryption scheme to recover the block u. Thus the client can get the real block. The privacy of our SPIR scheme is based on the hardness of LWE with binary error problem.
- **The VSPIR scheme using the probabilitic verification process** (see Fig. 1). Based on our proposed SPIR scheme, we use a probabilistic verification process [5] to construct our VSPIR scheme. The main idea of the probabilistic verification is very simple: a client samples a random input r and precomputes a specific function $F(r)$. He sends an input pair (x, r) to a

server in a random order, and wants to receive both $F(x)$ and $F(r)$ from the server. When receiving the answers from the server, the client checks the correctness of the response value $F(r)$; if it is the same as the precomputed $F(r)$, then the client accepts the response $F(x)$, and rejects otherwise. Because both x and r are independent and distributed identically, no malicious adversary can distinguish the real input x from the random input r and deceive with probability greater than 1/2. In our proposed VSPIR scheme, we do the similar process: the client generates a random vector $\mathbf{r} \in \{0,1\}^m$ to replace the u-th row in a matrix. Encrypt this matrix, then send the query message to the server and decode the responses R_j from the server. If the elements of the random vector corresponding to the index are the same, the elements of u-th row are the same. Then, the client accepts the received responses.

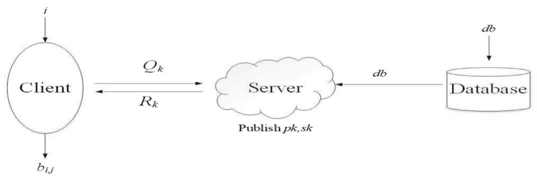

Fig. 1. Verifiable single-server PIR

For showing the merits of our proposal, we list the differences between our scheme and some other related scheme in Table 1. Specifically, our construction is essentially different from the SPIR construction proposed by Brakerski and Vaikuntanathan [3]. In our proposal, the client uses the encryption scheme with the additively homomorphic property to encrypt the index directly and the server responses the answer using the addictive homomorphically evaluate the database access function. However, in [3], the client encrypts the symmetric key using the FHE or SWHE scheme, then uses the encrypted symmetric key to encrypt the index, which can convert the symmetric ciphtertexts into homomorphic ciphertexts. The server uses the homomorphic ciphertexts homomorphically evaluate the database access function to retrieve an encryption of the answer. Moreover, in our SPIR construction, since using the encryption scheme based on the LWE with binary error assumption, the matrix multiplication operation in encryption scheme is equivalent to some matrix addition operation. The computational complexity can be slowed down to be $\mathcal{O}(\sqrt{tb})$.

1.5 Outline of Our Paper

The rest of this paper is organized as follows: in Sect. 2, after finishing notations used in this paper, we introduce the LWE problem and some definitions related to the VSPIR. In Sect. 3, we detail our proposed constructions for the SPIR scheme and VSPIR scheme, and then in Sect. 4 we analyze their performances.

In Sect. 5, we present some computer simulations for our proposals. Finally, in Sect. 6, we make some concluding remarks.

Table 1. Comparisons between the proposed scheme and some other related schemes.

	Zhang and Safavi-Naini [18]	Brakerski and Vaikuntanathan [3]	Vannet and Kunihiro [16]	Our VSPIR
Single-server	No	Yes	Yes	Yes
Verifiability	Yes	No	No	Yes
Assumption	d-SBDH	LWE	AGCD	LWE with binary error
Communication complexity	$\mathcal{O}(\kappa n^{1/\lfloor(2k-1)/t\rfloor})$	$\mathcal{O}(\log n + \kappa \mathsf{poly} \log(\kappa))$	$\mathcal{O}(\sqrt{n}\log n)$	$\mathcal{O}(c\sqrt{n})$
Computational complexity	$\mathcal{O}(\kappa n^{2/\lfloor(2k-1)/t\rfloor})$	$\mathcal{O}(\kappa^3)$	$\mathcal{O}(n\log n)$	$\mathcal{O}(c\sqrt{n})$

n: database size; k: server number related parameter; κ: security parameter; t: number of servers

2 Preliminaries

2.1 Notations

Before we present our scheme, we give some notations used in this paper. In this work, We denote vectors by bold lower-case letters $(\mathbf{x}, \mathbf{y}, \cdots)$, matrices by bold upper-case letters $(\mathbf{X}, \mathbf{Y}, \cdots)$. We denote a security parameter by $\kappa \in \mathbb{N}_+$. We denote the class of polynomial functions in κ by $\mathsf{poly}(\kappa)$, some fixed polynomial functions q in κ by $q = q(\kappa)$, and some unspecified negligible function in κ by $\mathsf{negl}(\kappa)$. We denote the transpose of $\mathbf{x}$ by $\mathbf{x}^T$. We consider the operation $x \overset{\$}{\leftarrow} \Psi$ as choosing x uniformly at random in a set Ψ. We use $\mathcal{D}$ to indicate a distribution over some finite set $\mathcal{S}$. We denote $x \overset{\$}{\leftarrow} \mathcal{D}$ that x is generated at random from the distribution.

2.2 Learning with Errors

The LWE problem was first introduced by Regev [15]. The formal definition can be as follows:

Definition 1 (LWE Problem [3]). *For security parameter κ, $n = n(\kappa)$, let $q = q(n)$ be an integer and error distribution $\chi = \chi(n)$ over $\mathbb{Z}_q$. Let $A_{s,\chi}$ be the distribution obtained by choosing a vector $\boldsymbol{a}$ from $\mathbb{Z}_q^n$ and an error term e from χ uniformly at random, and outputting $(\boldsymbol{a}, \langle a, s\rangle + e) \in (\mathbb{Z}_q^n \times \mathbb{Z}_q)$. The learning with errors problem $\mathsf{LWE}_{n,m,q,\chi}$ defined as follows: Given m independent instances from $A_{\boldsymbol{s},\chi}$, output $\boldsymbol{s}$ with non-negligible probability.*

The decision variant of the LWE problem, denoted $\mathsf{decision\text{-}LWE}_{n,m,q,\chi}$ is to distinguish the following two distributions: One is that sampling m instances

$(\boldsymbol{a}_i, \boldsymbol{b}_i)$ uniformly from $\mathbb{Z}_q^{n+1}$. The other one is that sampling m instances sampled according to $A_{\boldsymbol{s},\chi}$. The $\textsf{decision-LWE}_{n,m,q,\chi}$ assumption is that the $\textsf{decision-LWE}_{n,\kappa,q,\chi}$ problem is computationally infeasible.

Regev proved in [15] that given certain module q and Gaussian error distribution χ, $\textsf{LWE}_{n,\kappa,q,\chi}$ problem is as long as certain worst-case lattice problems which are hard to solve using a quantum algorithm. These reductions take χ to be the discretized versions of the Gaussian distribution which is B-bounded for an appropriate value B.

Definition 2 (B-Bounded Distributions [3]). *A distribution ensemble $\{\chi_n\}_{n\in N}$, supported over the integers, is called B-bounded if*

$$\Pr_{e\leftarrow\chi_n}[|e| > B] \leq \textsf{negl}(n).$$

The following theorem is the Regev's worst-case to average-case reduction for $\textsf{LWE}$:

Theorem 1 ([15]). *For $q = q(n) \in \mathbb{N}$ be a product of $q = \prod q_i$ such that for all i, $q_i = \textsf{poly}(n)$, and let $B \geq n$. There exists an efficiently sampleable B-bounded distribution χ such that if there is an efficient algorithm that solves the $\textsf{decision-LWE}_{n,q,\chi}$ problem, then there is an efficient quantum algorithm for solving $\tilde{O}(qn^{1.5}/B)$-apporoximate worst-case $\textsf{SIVP}$ and $\textsf{gapSVP}$.*

We refer the readers to [14,15] for the detailed and formal definitions of these lattice problems.

Definition 3 ($\textsf{LWE}$ with Binary Error Problem [10]). *Let n, q be positive integers, χ be a uniform distribution on $\{0, 1\}$ and $\boldsymbol{s} \overset{\$}{\leftarrow} \chi^n$ be a secret vector in $\{0, 1\}^n$. Let $A^{'}_{\boldsymbol{s},\chi}$ be the distribution obtained by choosing a vector $\boldsymbol{a} \in \mathbb{Z}_q^n$ uniformly at random and a noise term $e \overset{\$}{\leftarrow} \chi$, and outputting $(\boldsymbol{a}, \langle\boldsymbol{a}, \boldsymbol{s}\rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.*

$\textsf{LWE}$ with binary error problem is to recover $\boldsymbol{s}$ from m samples $(\boldsymbol{a}_i, \langle\boldsymbol{a}_i, \boldsymbol{s}_i\rangle + e_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.

The decision variant of the $\textsf{LWE}$ with binary error problem is to distinguish with non-negligible advantage m samples chosen according to $A^{'}_{\boldsymbol{s},\chi}$, from m samples chosen according to the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

Theorem 2 ([13]). *For any integers n and $m = n\cdot(1+\Omega(1/\log n))$, and all sufficiently large polynomially bounded prime modulus $q \geq n^{\mathcal{O}(1)}$, solving $\textsf{LWE}_{n,m,q}$ with uniformly random binary errors (i,e, in $\{0,1\}$) is at least as hard as approximating lattice in the worst case on $\Theta(n/\log n)$-dimensional lattices within a factor $\gamma = \tilde{\mathcal{O}}(\sqrt{n}\cdot q)$.*

Theorem 2 shows that for the $\textsf{LWE}$ problem, it remains hard even when the errors are small (e.g, uniformly random from $\{0, 1\}$). Most cryptographic constructions are based on the $\textsf{LWE}$ problem where secret and error are identically distributed [10]. Using the search-to-decision reduction of [13], Peikert et al. proved that $\textsf{decision-LWE}_{n,m,q}$ with binary error has the similar hardness of $\textsf{LWE}_{n,m,q}$ with binary error.

2.3 The GHV-Type Encryption Scheme

The basis of the GHV scheme [7] is a trapdoor sampling algorithm [8]. The trapdoor sampling procedure generates a matrix $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$ (that is within negligible statistical distance of uniform), together with an invertible matrix $\mathbf{T} \in \mathbb{Z}^{m\times m}$ with small entries such that $\mathbf{T} \cdot \mathbf{A} = 0 (\text{mod } q)$.

The trapdoor can be used to solve the LWE problem relative to $\mathbf{A}$. This is done as follows: $\mathbf{Ty} = \mathbf{T}(\mathbf{As} + \mathbf{x}) = \mathbf{Tx}(\text{mod } q)$. Multiplying $\mathbf{T}^{-1}$ gives us $\mathbf{x}$. There is a probabilistic polynomial-time (PPT) algorithm TrapDoor that, on input 1^κ, positive integer $q \geq 2$, and a poly(n)-bounded positive integer $m \geq 5n \log q$, output matrices $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$ and $\mathbf{T} \in \mathbb{Z}^{m\times m}$ where the Euclidean norm of each rows is at least $20n \log q$ [1].

The GHV-type encryption scheme [7] is defined by a triple PPT algorithm GHV = (GHV.KeyGen, GHV.Enc, GHV.Dec):

- GHV.KeyGen$(1^\kappa) \to (pk, sk)$: On input the 1^κ, let $n = \kappa$, run the trapdoor sampling algorithm to obtain a matrix $\mathbf{A} \in Z_q^{m\times n}$ together with a trapdoor matrix $\mathbf{T} \in Z^{m\times m}$, i.e., $(\mathbf{A},\mathbf{T}) \leftarrow$TrapDoor$(1^n, q, m)$. The public key pk is $\mathbf{A}$ and the secret key sk is $\mathbf{T}$.
- GHV.Enc$_{pk}(\mathbf{M}) \to \mathbf{C}$: To encrypt the binary message $\mathbf{M} \in \{0, 1\}^{m\times m}$, choose a uniformly random matrix $\mathbf{S} \overset{\$}{\leftarrow} \mathbb{Z}_q^{n\times m}$ and an "error matrix" $\mathbf{X} \overset{\$}{\leftarrow} \chi^{m\times m}$. Output the ciphertext $\mathbf{C} \leftarrow \mathbf{AS} + 2\mathbf{X} + \mathbf{M}(\text{mod } q)$ where $2\mathbf{X}$ means multiplying each entry of the matrix $\mathbf{X}$ by 2.
- GHV.Dec$_{sk}(\mathbf{C}) \to \mathbf{M}$: Set $\mathbf{E} \leftarrow \mathbf{TCT}^T(\text{mod } q)$, and then output $\mathbf{B} \leftarrow \mathbf{T}^{-1}$ $\mathbf{E}(\mathbf{T}^T)^{-1} \mod 2$.

2.4 Formal Definitions About VSPIR

Definition 4 (VSPIR). *The VSPIR scheme consists of a database owner server S,and a client C. S has the database db* $= (db_1, \cdots, db_{tb})$. *C owns an index* $i \in [tb]$ *and wants to recover the* db_i *from the clouds, keeping the i secret. The VSPIR scheme is defined by five PPT algorithms VSPIR = (VSP.Setup, VSP.Query, VSP.Challenge, VSP.Response, VSP.Verify):*

1. *VSP.Setup*$(1^\kappa) \to (pk, sk)$ *: On input* 1^κ*, output the public key pk and secret key sk.*
2. *VSP.Query*$_{sk}(i) \to (Q, aux)$ *: On input a private key sk and an index* $i \in [tb]$*, output a query Q along with auxiliary information aux.*
3. *VSP.Challenge*$_{sk}(i, \kappa) \to L$*: On input* κ*, the index i and sk output the challenge message L.*
4. *VSP.Response*$_{pk}(Q, db, L) \to R$*: On input a public key pk, the query Q, database db and the challenge L. Output the response message R.*
5. *VSP.Verify*$_{sk}(Q, R, aux) \to \{db_i, \perp\}$ *: On input sk, Q, the response message R, and the auxiliary aux. Output the* db_i *or* $\perp$.

The server S who owns the database is responsible to set up the system. To set up the system, S runs VSP.Setup to obtain (pk, sk) in the off-line stage.

Then pk is published or sent to server, the database db is given to the cloud, and the sk is kept private by client. To retrieve db_i, C runs VSP.Query to compute (Q, aux) and sends the query message Q to the cloud S. Upon receiving Q, S runs VSP.Response and replies with the responses message R. To verify the responses, C runs VSP.Challenge to generate a challenge message L and runs VSP.Verify to verify the responding messages and compute db_i if the algorithm VSP.Verify does not output the failure message.

Now, we present some formal properties of VSPIR, these definitions are based on the previous work [3,16,18].

Definition 5 (Correctness). *The VSPIR scheme is convinced to be correct if the verify algorithm always computes the correct value of db_i when the server gives the correct response. Formally, for κ, database db, let $\mathsf{VSP.Setup}(1^\kappa) \rightarrow (pk, sk)$, for any query index $i \in [tb]$, let $\mathsf{VSP.Query}_{sk}(i) \rightarrow (Q, aux)$ and $\mathsf{VSP.Response}_{pk}(Q,\ db,\ L) \rightarrow R_i$, it holds that*

$$\Pr[\mathsf{VSP.Verify}_{sk}(Q, R, aux) \neq db_i] \leq \mathsf{negl}(\kappa)$$

if the verify algorithm does not compute the failure message.

Definition 6 (Privacy). *The scheme VSPIR is convinced to be private if the adversary can not learn any information about i. Namely, for two queries i_1,$i_2 \in [tb]$ it can computationally distinguish $\mathsf{VSP.Query}_{sk}(i_1)$ from $\mathsf{VSP.Query}_{sk}(i_2)$ with negligible probability. Formally, let κ be a security parameter for an adversary $\mathcal{A}$ running in polynomial time and asking polynomially many queries, it holds that*

$$\Pr[\mathcal{A}(\mathsf{VSP.Query}_{sk}(i_1))] - \Pr[\mathcal{A}(\mathsf{VSP.Query}_{sk}(i_2))] \leq \mathsf{negl}(\kappa).$$

Definition 7 (Security). *The scheme VSPIR is convinced to be secure if PPT adversary can deceive the client into obtaining an incorrect value of db_i with negligible probability. We can consider the behavior of $\mathcal{A}$ in a number of $Game_0$, $Game_1$, $Game_2$ as defined below:*

1. *$Game_{0.}$ The challenger generates $\mathsf{VSP.Setup}(1^\kappa) \rightarrow (pk, sk)$ and then publishes pk to $\mathcal{A}$. $\mathcal{A}$ owns the database db and every time $\mathcal{A}$ chooses an index i the challenger will reply corresponding query message Q.*
2. *$Game_{1.}$ $\mathcal{A}$ picks a specific index i and sends it to the challenger, the challenger responses $\mathsf{VSP.Query}_{sk}(i) \rightarrow (Q, aux)$. To verify the db_i, challenger runs $\mathsf{VSP.Challenge}_{sk}(i, \kappa) \rightarrow C$.Then $\mathcal{A}$ runs $\mathsf{VSP.Respon\text{-}\ se}_{pk}$ $(Q, db, C) \rightarrow R$ to response the challenger.*
3. *$Game_{2.}$ $\mathcal{A}$ wins if $\mathsf{VSP.Verify}_{sk}(Q, R, aux) \notin \{db_i, \perp\}$.*

In the security $Game_2$, $\mathcal{A}$ can deceive the client into reconstructing an incorrect value of db_i even if it can choose the index of database freely with negligible probability. Thus, the security of a VSPIR scheme defined above allows the client to recover the correct block that he wants to obtain from the database

Definition 8 (Communication Complexity). *The communication complexity of a scheme is defined as the number of bits being exchanged to transfer a single database element excluding the setup phase.*

Definition 9 (Index Mapping Function). *We define an index mapping function which maps the index u to an vector matrix. It takes as input an index u in some scope and output an index vector: $\boldsymbol{\delta}_{i,u} \leftarrow E(u)$. where the u-th element of the vector is 1, the others are 0.*

3 Our Constructions

In this section, we demonstrate our scheme in a gradual manner. We first present our variant of the GHV-type encryption scheme and an SPIR using this variant. After that, based on the proposed SPIR, we give a VSPIR construction under the malicious server model.

3.1 A Variant of the GHV-Type Encryption Scheme

In GHV scheme [7], it can encrypt a matrix of m^2 elements in time $\tilde{O}(m^3)$. To reduce the computational complexity, we consider the LWE with binary error assumption. Luckly, previous work [10,13] has proved the hardness of the LWE with binary error problem.

For ease of presentation, we focus below on the case of encrypting binary vectors for better use in our SPIR scheme. The extension for encrypting matrices with lower computational complexity comparable to GHV scheme is straightforward. Our variant of the GHV-type encryption scheme VGHV = ($\mathsf{VG.KeyGen}$, $\mathsf{VG.Enc}$, $\mathsf{VG.Dec}$) is a triple of PPT algorithms as follows:

- $\mathsf{VG.KeyGen}(1^\kappa) \rightarrow (pk, sk)$: The algorithm is the same as the algorithm in GHV scheme. The public key $pk = \mathbf{A} \in \mathbb{Z}_q^{m\times n}$ and the secret key $sk = \mathbf{T} \in \mathbb{Z}^{m\times m}$.
- $\mathsf{VG.Enc}_{pk}(\mathbf{m}) \rightarrow \mathbf{c}$: To encrypt $\mathbf{m} \in \{0,1\}^m$, choose a uniformly random vector $\mathbf{s} \in \{0,1\}^n$ and a uniformly random error vector $\mathbf{x} \in \{0,1\}^m$. Output the ciphertext $\mathbf{c} \leftarrow \mathbf{A}\mathbf{s} + 2\mathbf{x} + \mathbf{m} (\text{mod } q)$ where $2\mathbf{x}$ means multiplying each entry of the vector $\mathbf{x}$ by 2.
- $\mathsf{VG.Dec}_{sk}(\mathbf{c}) \rightarrow \mathbf{m}$: Set $\mathbf{e} \leftarrow \mathbf{T}\mathbf{c} \mod q$, and then output $\mathbf{m} \leftarrow \mathbf{T}^{-1}\ \mathbf{e} \mod 2$.

For the decryption algorithm, recall that $\mathbf{T}\cdot\mathbf{A} = 0 (\text{mod } q)$ and therefore $\mathbf{T}\mathbf{c} = \mathbf{T}(2\mathbf{x}+\mathbf{m})(\text{mod } q)$. If in addition all the entries of $\mathbf{T}(2\mathbf{x}+\mathbf{m})$ are smaller than q then we also have the equality over the integers $\mathbf{e} = (\mathbf{T}\mathbf{c}(\text{mod } q)) = \mathbf{T}(2\mathbf{x}+\mathbf{m})(\text{mod } q)$, and hence $\mathbf{T}^{-1}\mathbf{e} = \mathbf{m}(\text{mod } 2)$. We have the correct decryption when all the entries of $\mathbf{T}(2\mathbf{x}+\mathbf{m})$ are smaller than $2/q$.

Additional Homomorphic Operation. Given two ciphertexts $\mathbf{c}_1$, $\mathbf{c}_2$ that decrypt to $\mathbf{m}_1$, $\mathbf{m}_2$. Let $\mathbf{c} = \mathbf{c}_1 + \mathbf{c}_2$. For addition, we have $\mathbf{c} = \mathbf{A}(\mathbf{s}_1 + \mathbf{s}_2) + 2(\mathbf{x}_1 + \mathbf{x}_2) + \mathbf{m}_1 + \mathbf{m}_2$ which can be decrypted as $\mathbf{m}_1 + \mathbf{m}_2$ when all entries in $\mathbf{T}(2(\mathbf{x}_1 + \mathbf{x}_2) + \mathbf{m}_1 + \mathbf{m}_2)$ are smaller than $q/2$.

Theorem 3 *Any distinguishing algorithm with advantage ϵ against the CPA privacy*[1] *of the scheme can be converted to a distinguisher against* $\mathsf{decision\text{-}LWE}_{m,n,q}$ *with binary error with roughly the same advantage at least* $\epsilon/2\,m$.

Proof. See Appendix D for the proof.

We can use the above variant of the GHV scheme to encrypt the binary matrices by setting uniformly random $\mathbf{S} \in \{0,1\}^{n\times m}$ and uniformly random "error matrix" $\mathbf{X} \in \{0,1\}^{m\times m}$. Specifically, we call this variant of the GHV scheme as MVGHV. Note that, our MVGHV is more efficient than the original GHV scheme: MVGHV takes time $\mathcal{O}(m \cdot n)$ to encrypt a matrix of m^2 elements comparing with $\tilde{\mathcal{O}}(m^3)$ in GHV scheme. The CPA privacy of MVGHV scheme is based on the LWE with binary error using the proof algorithm in [7].

Theorem 4. *For the parameter $n = n(\kappa)$ and $c = c(n) > 0$, let $q > 8n^{3c}$, $m = \lfloor 5n\log q\rfloor$, then the encryption scheme from above with parameters n,m,q supports n^c additions.*

Proof. See Appendix A for the proof.

Theorem 4 shows that the number of LWE with binary error samples $m = \mathcal{O}(n)$ is linear. For selection about m, it can be satisfied by taking $m = 5n\log q$ for fixed q.

3.2 Our SPIR Scheme

Now we introduce our SPIR scheme. We redefine the database $(db_1, \cdots, db_{tb})$, $db_i \in \{0,1\}$: assume the database can be split into nb blocks of mw words of bb bits each, such that $nb \cdot mw \cdot bb = tb$ (tb is the size of database). Namely, the nb is the count of blocks, and mw for words per block and bb for bits per word. If nb cannot be decomposed in this way, pad the database with several extra bits to make it that. We denote the database by a 2-dimensional array of words where each word is marked by two coordinates. Then we obtain the database $b = \{b_{i,j} | 1 \le i \le nb, 1 \le j \le mw\}$, the total bit size of the database is tb. First we assume that every word is a single bit, clearly $mw = 1$.

Assume that the client C wants to recover the block u that is consisted of $b_{u,j}$. We present the SPIR with four PPT algorithms SPIR = (SP.Setup, SP.Query, SP.Response, SP.Dec):

- SP.Setup$(1^\kappa) \to (pk, sk)$: This algorithm is to set up the system and generate the public key pk and the secret key sk. On input κ, Run the VG.KeyGen to obtain the public key $pk = \mathbf{A} \in \mathbb{Z}_q^{m\times n}$, and the secret key $sk = \mathbf{T} \in \mathbb{Z}^{m\times m}$. The pk is published to S, and the sk is kept secretly in C.

[1] The notation of CPA privacy is equivalent to the formal notation of CPA security.

- $\mathsf{SP.Query}_{pk}(u) \to Q$: This algorithm for C is to obtain the query string. On input the public key pk and index u, compute the function $E(u)$ to obtain the index vector $\boldsymbol{\delta}_{i,u} \in \{0,1\}^{nb}$, and then spit it to $m_c = \lceil nb/m \rceil$ vectors, if $\boldsymbol{\delta}$ cannot be decomposed in this way, pad the last vector with several 0 elements. Encrypt these vectors in order. Run the algorithm $\mathsf{VGHV.Enc}_{pk}(\mathbf{m}_c) \to \mathbf{c}_c$, $c \in [m_c]$. The query message Q is these ordered vectors $\mathbf{c}_c$.
- $\mathsf{SP.Response}(b, Q) \to R$: This algorithm for S is to compute the responses. On input the query message Q and the database b, compute the responses for every j from 1 to mw: $\mathbf{r}_j = \sum_{c=1}^{m_c} b_{m(c-1)+i,j}\mathbf{c}_c$ for i from 1 to m. According to the homomorphism, multiply $b_{i,j}$ by $\mathbf{c}$ corresponds to the multiplication of $b_{i,j}$ and $\boldsymbol{\delta}$. Thus $\mathbf{r}_j$ is the homomorphic sum of $b_{i,j}\mathbf{m}_c$. The element of vector $\boldsymbol{\delta}_{i,u}$ is 1 where $i = u$, otherwise 0. The $\mathbf{r}_j$ is the ciphertext of block b_u. The response message R is these vectors $\mathbf{r}_j$.
- $\mathsf{SP.Dec}_{sk}(R) \to b_{u,j}$: This algorithm for S is to recover the block that C wants. On input the secret key sk and responses R, run $\mathsf{VG.Dec}_{sk}(\mathbf{r}_j)$ to obtain the block u.

Multi-Words SPIR Scheme. In the above scheme, we assume that the word is a single bit. We can easily modify it to recover multi-words scheme. Instead of computing $\mathbf{As} + \mathbf{2e}$, we compute $\mathbf{As} + 2^{mw}\mathbf{e}$, $b_{u,j}\mathbf{e}_j \leftarrow b_{u,j}\mathbf{T}_j(2\mathbf{x}_j + \boldsymbol{\delta}_j) \mod q$, hence $b_{u,j}\mathbf{T}_j^{-1}\mathbf{e} \mod 2^{mw} = b_{u,j}\boldsymbol{\delta}_j$. Since q has to be large for security reasons and the noise only progresses linearly when processing the database, we can afford to start with a fairly large noise. We can utilize the same trick to obtain the "multi-words matrix retrieval".

3.3 Our VSPIR Scheme

Before introducing our VSPIR scheme,we show the probabilistic verification process used in the work [5]. C delegates some computation F to an untrusted server S, and C wants to verify the response from S. Assume that C can precompute $F(x)$, we can define it as three procedures:

- *Setup.* Input κ, the delegated computation function $F : \{0,1\}^n \to \{0,1\}^m$, and the value $x \in \{0,1\}^n$.
- *Precomputation.* C samples a random input r, computes $w = F(r)$, and stores (r, w) as secret state.
- *Delegation and Verification.* C has an input $m \in \{0,1\}^n$. C sets $r_0 = r$ and $r_1 = m$, then samples a random bit $b \in \{0,1\}$, and sends the pair (r_b, r_{1-b}) to S. S computes and sends $(z_0, z_1) = (F(r_0), F(r_1))$ to C. Then C accepts and recovers the response z_{1-b} if $w = z_b$.

From the work in [5], we can find that since x and r are independent and identically distributed, no malicious adversary can cheat C successfully with non-negligible probability. Our VSPIR scheme employs the similar probabilistic verification procedures as above. The main difference of probabilistic verification procedure between our VSPIR and the proposal in [5] is that we use a random vector to mark the result that we want instead of precomputating. Now we detail the description of our VSPIR scheme $\mathsf{VSPIR} = (\mathsf{VSP.Setup}, \mathsf{VSP.Challenge}, \mathsf{VSP.Response}, \mathsf{VSP.Verify})$:

- $\mathsf{VSP.Setup}(1^\kappa) \rightarrow (sk, pk)$. On input 1^κ, run $\mathsf{SP.Setup}$ and output (sk, pk).
- $\mathsf{VSP.Challenge}_{sk}(u) \rightarrow Q$. On input the index u and sk, pick up a random vector $\mathbf{v} \in \{0,1\}^m$, if the d-th element of $\mathbf{v}v_d$ is 1, then run index mapping function $\mathbf{v}_{i,u} \leftarrow E(u)$ otherwise generate a 0 vector with the same dimension. Then combine these vectors into a matrix $\mathbf{I}$. Run the encryption algorithm in MVGHV to encrypt the matrix using the $\mathsf{SP.Query}$ way to obtain the query Q.
- $\mathsf{VSP.Response}(b, Q) \rightarrow R_k$. On input database b and query messages Q, compute $\mathsf{SP.Response}(b, Q)$ to obtain the responses R_j.
- $\mathsf{VSP.Verify}_{sk}(Q,\ R_j) \rightarrow \{b_u, \perp\}$. On input Q and the responses messages R_j, run the decryption algorithm in MVGHV and make sure whether the decryption result is our expectation. Accept and output the recovers if when the elements of the random vector's corresponding index are the same,the elements of u-th row in the matrices $b_{u,j}\mathbf{I}$ are the same as $b_{u,j}$. Otherwise, reject and output $\perp$.

4 Performance Analysis

In this section, we first analyze correctness, privacy and security of the proposed SPIR and VSPIR scheme. After that, we present communication complexity and computational complexity of our proposals.

4.1 Correctness, Privacy and Security

Theorem 5 (Correctness). *If the proposed MVGHV encryption scheme holds the homomorphic property for supporting polynomially many additions,then our VSPIR scheme can computes the correct retrieval information when server gives the correct response.*

Proof. See Appendix B for the proof.

Theorem 6 (Privacy). *If any distinguishing algorithm can distinguish two queries for distinct bits of database with probability at least $1/2 + \epsilon/2$, then the distinguisher can break the privacy of our VSPIR with probability $(1 + \epsilon)/2$.*

Proof. See Appendix C for the proof.

Theorem 7 (Security). *Our VSPIR scheme is convinced to be secure if no PPT adversary can deceive the client into obtaining an incorrect value responded from the server with non-negligible probability.*

Proof. See Appendix D for the proof.

4.2 Complexity

In this section, we analyze the communication complexity and computational complexity of our protocols. Recall our schemes, the public key is sent only once, it is independent of the database and the query, and it can be used for many queries. Therefore it is customary to analyze such schemes in the public key model where sending the public key does not count towards the communication complexity.

For SPIR scheme, to encrypt the query vectors, the size of ciphertext in our encryption algorithm is composed of $\lceil nb/m \rceil$ vectors whose size is $\lceil nb/m \rceil m \log q$. Then the response size is $m \log q$. When $nb = mw = \sqrt{tb}$, for fixed security parameter, the communication complexity is $\mathcal{O}(c^{'}\sqrt{tb})$. For VSPIR scheme, the total size of one round transform messages comes to $(m^2 + \lceil nb/m \rceil m^2) \log q$ and the communication complexity is $\mathcal{O}(c\sqrt{tb})$ (c and $c^{'}$ are constant).

The communication complexity is not changed in the process for multiple bits recovery, but we now can retrieve a block of mw bb bits. Furthermore, when $nb = mw = \sqrt{tb/bb}$ the communication complexity is $\mathcal{O}(c\sqrt{tb/bb})$ per an index recovered.

Now let us look at the computational complexity. To set up the system, the key generation algorithm is executed once and takes time $\mathcal{O}(1)$. If not considering any optimization algorithm, to encrypt the vector index it takes about $\lceil nb/m \rceil m$ operations and to encrypt the matrix index it takes about $\lceil nb/m \rceil m \cdot n$. When $nb = mw = \sqrt{tb}$, for fixed security parameter, the computational complexity is $\mathcal{O}(\sqrt{tb})$ and $\mathcal{O}(c\sqrt{tb})$, respectively.

5 Computer Implementations

In this section, we made a straightforward implementation of our VSPIR scheme without aiming for high levels of optimization. The timings were performed on a

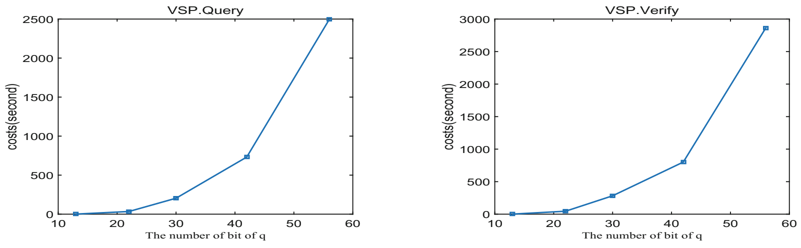

Fig. 2. The trends of the client's costs in our VSPIR scheme

Table 2. The client's costs in our VSPIR scheme (second).

(ψ, n)	RVSP.Query	RVSP.Verify
(13, 8)	1.012	1.010
(22, 51)	34.236	45.236
(30, 269)	204.415	282.754
(42, 531)	731.950	801.481
(56, 1563)	2494.514	2864.157

ψ: the number of bit of q

2013 ASUS (Intel(R) Core(TM) i5-3230M, 2 hyperthreaded cores at 2.60 GHz, 8 GB RAM at 1.600 GHz), on Windows (Windows 10 Home, x64 64). Our implementations are single-threaded. We used NTL for operations over $\mathbb{Z}_q$, matrix operations, and big number operations. We implement the algorithm $\mathsf{VSP.Query}$ and the decryption part of $\mathsf{VSP.Verify}$. To simplify the operation, we focus on a matrix. For showing the cost of the proposed VSPIR scheme, we list the time costs in Table 2 when choosing different parameters and draw the trends of the proposed VSPIR in the Fig. 2. Note that, second can be denoted by s.

Now we consider some real scenarios. As showed in [17], the maximum bandwidth of common Internet access technologies such as the Wireless 802.11 g is 54Mbit/s, Fast Ethernet is 100Mbit/s, OC12 is 622Mbit/s. In these scenarios, the communication complexity of the server and the client can be asymmetrically negligible. Let the database be 1G bits, we set the bit of $q = 30$, $n = 269$, $nb = \sqrt{tb}$. Assume that the upload speed and download speed are all 20Mbit/s, now we can roughly compute the time cost of one round query and the response is about 31 s. The probabilistic verification process can increase the time costs slightly.

6 Conclusions

In this paper, we have proposed an efficient SPIR scheme based on the LWE with binary error assumption and a VSPIR scheme using the probabilistic verification that can work under the malicious server model. Compared with previous works, our scheme is the first practical VSPIR scheme under the malicious server model that we know of. Specifically, our VSPIR scheme has communication complexity $\mathcal{O}(c\sqrt{tb})$ that is smaller than the communication complexity of the proposal in [18].

Acknowledgements. This work was supported in part by the National Natural Science Foundation of China under Grant 61302161, in part by the Doctoral Fund, Ministry of Education, China, under Grant 20130181120076. The corresponding author of this research Liang Zhao is supported by the International Visiting Program for Excellent Young Scholars of SCU.

A PROOF OF THEOREM 3

Proof (sketch). Let $\mathcal{A}$ be a CPA-adversary that distinguishes between encryptions of messages of its choice with advantage ϵ. First, We construct a distinguisher $\mathcal{D}$ with advantage at least $\epsilon/2$ between the two distributions: $\{(\mathbf{A}, \mathbf{As} + \mathbf{x}) : \mathbf{A} \leftarrow \mathbb{Z}_q^{m\times n}, \mathbf{s} \in \{0,1\}^n, \mathbf{x} \in \{0,1\}^m\}$ and $\{\mathsf{Unif}(\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m)\}$. The distinguisher $\mathcal{D}$ takes as input $(\mathbf{A}, \mathbf{c})$, and runs the adversary $\mathcal{A}$ with $\mathbf{A}$ as the public key. Upon receiving message $\mathbf{m}_0, \mathbf{m}_1$ from the adversary, $\mathcal{D}$ chooses at random $i \in \{0,1\}$, returns the challenge ciphertext $2\mathbf{c} + \mathbf{m} \mod q$, then outputs 1 if the adversary $\mathcal{A}$ guesses the right i, and 0 otherwise.

On the one hand, if $\mathbf{c}$ is uniformly random matrix then the challenge ciphertext is also uniformly random, regardless of the choice of i. Hence in this case $\mathcal{D}$ outputs 1 with probability at most 1/2. On the other hand, if $\mathbf{c} = \mathbf{As}+\mathbf{x} (\mathrm{mod}\ q)$, then the challenge ciphertext is $2\mathbf{c}+\mathbf{m}=\mathbf{A}\mathbf{s}^{'}+2\mathbf{x}+\mathbf{m}(\text{mod } q)$. By assumption $\mathcal{A}$ will guess the right i with probability $(1+\epsilon/2)$. Finally, a standard hybrid argument can be used to convert the distinguisher $\mathcal{D}$ from above to a $\mathsf{decision\text{-}LWE}_{m,n,q}$ with binary error distinguisher with advantage $\epsilon/2\text{m}$.

B PROOF OF THEOREM 4

Proof. Let vector $\mathbf{c}=\sum_{i=1}^{\ell}(\mathbf{A}\mathbf{s}_i+2\mathbf{x}_i+\mathbf{m}_i)$ be obtained by adding $\ell \leq n^c$ ciphertexts. Recall that every row of $\mathbf{T}$ has Euclidean norm at most $20n\log q$. Then every entry of $\sum_{i=1}^{\ell}\mathbf{T}\mathbf{x}_i$ is at most $20\ell n\log q$. All the $\mathbf{m}_i^{'}$s are binary so each entry of $\mathbf{T}\mathbf{m}$ is at most $20\ell n\log q$. Hence the each entry in $\mathbf{T}(2\mathbf{x}+\mathbf{m})$ is bounded by $40\ell n\log q < 4n^{3c} < q/2$ for some q. Now we can decrypt $\mathbf{c}$ to recover the correct value.

C PROOF OF THEOREM 5

Proof. On the one hand, our proposed similar probabilistic verification procedures provide the correctness of the response message from server. On the other hand, as introduced in the preliminaries section, the GHV scheme is correct regard to additive homomorphic operation, our MVGHV scheme follows the same property. Then our VSPIR using the MVGHV scheme has the correctness when the server provides the correct response.

D PROOF OF THEOREM 6

Proof (sketch). Let $\mathcal{A}$ be a CPA-adversary that distinguishes between encryptions of queries of its choice with advantage ϵ, we first construct a distinguisher $\mathcal{D}$ with advantage at least $\epsilon/2$ between two queries: the query the client wants and a query chosen randomly. The distinguisher $\mathcal{D}$ takes as input a pair of matrices ($\mathbf{A}$, $\mathbf{C}$), and runs the adversary $\mathcal{A}$ with $\mathbf{A}$ as the public key. Upon receiving message $\mathbf{B}_0$,$\mathbf{B}_1$ from adversary, $\mathcal{D}$ chooses at random $i \in \{0,1\}$, returns the challenge ciphertext encrypted by MVGHV scheme, then outputs 1 if the adversary $\mathcal{A}$ guesses the right i, and 0 otherwise. On the one hand, if $\mathbf{C}$ is the random query ciphertext, then $\mathbf{C}$ is equivalent to be a uniformly distribution. In this case $\mathcal{D}$ outputs 1 with probability at most 1/2. On the other hand, If $\mathbf{C}$ encrypts the query the client wants, by assumption $\mathcal{A}$ will guess the right i with probability $(1+\epsilon)/2$. Then the privacy of our VSPIR is based on the $\mathsf{decision\text{-}LWE}$ with binary error assumption.

E PROOF OF THEOREM 7

Proof. To verify the correctness of the response, we sample a random vector replace the index matrix. If an algorithm can break the privacy of our scheme with probability $\mathsf{negl}(\kappa)$, the algorithm can distinguish queries for block X_1 and block X_2. Then the adversary can deceive the client into obtaining an incorrect value with probability $\mathsf{negl}(\kappa)/2^m$.

References

1. Alwen, J., Peiket, C.: Generating shorter bases for hard random lattices. In: STACS, pp. 75–86 (2009)
2. Beimel, A. Ishai, Y.: Private information retrieval: a primer. www.cs.bgu.ac.il/beimel/Papers/PIRsurvey.ps
3. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: FOCS, pp. 97–106 (2001)
4. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private Information Retrieval. In: FOCS, pp. 41–50 (1995)
5. Chung, K.-M., Kalai, Y., Vadhan, S.: Improved delegation of computation using fully homomorphic encryption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 483–501. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_26
6. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_2
7. Gentry, C., Halevi, S., Vaikuntanathan, V.: A simple BGN-type cryptosystem from LWE. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 506–522. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_26
8. Gentry, C. Peikert, C. Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC, pp. 197–206. ACM (2008)
9. Goldberg, I.: Improving the robustness of private information retrieval. In: 2007 IEEE Symposium on Security and Privacy, pp. 131–148 (2007)
10. Johannes, A., Buchmann, F.G., Rachel, P., Thomas W.: On the hardness of LWE with binary error: revisiting the hybrid lattice-reduction and meet-in-the-middle attack. In: AFRICACRYPT, pp. 24–43 (2016)
11. Kumar, M., S., Sarkar, P.: Symmetrically private information. In: INDOCRYPT, pp. 225–236 (2000)
12. Micciancio, D., Mol, P.: Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 465–484. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_26
13. Micciancio, D., Peikert, C.: Hardness of SIS and LWE with small parameters. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 21–39. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_2
14. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem. In: STOC, pp. 333–342 (2009)
15. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM **56**(6), 34 (2009). Preliminary version in STOC 2005
16. Vannet, T., Kunihiro, N.: Private information retrieval with preprocessing based on the approximate GCD problem. In: Dunkelman, O., Keliher, L. (eds.) SAC 2015. LNCS, vol. 9566, pp. 227–240. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31301-6_14
17. Wikipedia. https://en.wikipedia.org/wiki/Bandwidthx
18. Zhang, L.F., Safavi-Naini, R.: Verifiable multi-server private information retrieval. In: Boureanu, I., Owesarski, P., Vaudenay, S. (eds.) ACNS 2014. LNCS, vol. 8479, pp. 62–79. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07536-5_5