Information-Theoretic Random-Index PIR

Sebastian Kolby
Aarhus University, Denmark

Lawrence Roy
Aarhus University, Denmark

Jure Sternad
Aarhus University, Denmark

Sophia Yakoubov
Aarhus University, Denmark

Abstract

A Private Information Retrieval (PIR) protocol allows a client to learn the ith row of a database held by one or more servers, without revealing i to the servers. A Random-Index PIR (RPIR) protocol, introduced by Gentry *et al.* (TCC 2021), is a PIR protocol where, instead of being chosen by the client, i is random. This has applications in e.g. anonymous committee selection. Both PIR and RPIR protocols are interesting only if the communication complexity is smaller than the database size; otherwise, the trivial solution where the servers send the entire database suffices.

Unlike PIR, where the client must send at least one message (to encode information about i), RPIR can be executed in a single round of server-to-client communication. In this paper, we study such one-round, information-theoretic RPIR protocols. The only known construction in this setting is SimpleMSRPIR (Gentry *et al.*), which requires the servers to communicate approximately $\frac{N}{2}$ bits, N being the database size. We show an $\Omega(\sqrt{N})$ lower bound on communication complexity for one-round two-server information-theoretic RPIR, and a sublinear upper bound. Finally, we show how to use a sublinear amount of database-independent correlated randomness among multiple servers to get near-optimal online communication complexity (the size of one row plus the size of one index description per server).

2012 ACM Subject Classification Security and privacy → Information-theoretic techniques

Keywords and phrases Private information retrieval, Multi-server, Lower bounds

Digital Object Identifier 10.4230/LIPIcs.ITC.2025.5

Funding Danish Independent Research Council under Grant-ID DFF-2032-00122B and DFF-2064-00016B (YOSO)

Acknowledgements Supported by the Danish Independent Research Council under Grant-ID DFF-2032-00122B and DFF-2064-00016B (YOSO).

1 Introduction

PIR (private information retrieval [4]) enables a client to retrieve the ith row of a database from one or more servers in such a way that i remains unknown to the server(s). The trivial solution involves the server(s) sending the client the entire database; however, it is desirable to bring the communication complexity down so that it is strictly smaller than the size of the database.

RPIR (random-index PIR [7]) is a relaxation of PIR, where instead of learning a database row of her choice, the client learns a random row. Just like PIR, RPIR guarantees that the index i of that row is unknown to the server(s). Any PIR construction can be converted to a RPIR construction, simply by having the client choose the index i at random. However, while PIR constructions require at least one message from the client (to communicate something about i) followed by at least one message from the server(s) (to communicate something about the database), not all RPIR constructions require the client to communicate.

Random-index PIR constructions that only take one round of server-client communication are termed “non-interactive”. Gentry *et al.* [7] show how to build one-server non-interactive RPIR from homomorphic encryption. They also show how to build two-server non-interactive information-theoretic RPIR (which they dub SimpleMSRPIR), the communication complexity of which is $\frac{dw}{2} + \log(d)$ (where the database contains d rows each of size w).

Our Contributions

In this paper, we take steps towards understanding the communication complexity of non-interactive information-theoretic RPIR. We prove that non-interactive two-server information-theoretic RPIR must have communication complexity at least $O(\sqrt{d})w$.

On the positive side, we give an upper bound with communication complexity

$$\frac{dw}{d^{\frac{\log\log(d)}{\log(d)}}} + \frac{dw}{\frac{\log(d)}{\log\log d} + 1} + \Theta(d\log(d))$$

bits, showing that the communication complexity can in fact be sub-linear in the database size dw (when $w = \omega(\log(d))$). [1]

Finally, we describe a construction that uses a sublinear amount of correlated randomness, but gets the near-optimal online communication complexity of $n\log(d)w$ (where n is the number of servers, d is the number of database rows, and w is the size of each row).

Related Work

Information-theoretic Private Information Retrieval (PIR) was first introduced by Chor et al. [4], who provided a general construction for the case of n servers, achieving a total communication complexity of $O(d^{\frac{1}{n}})$. Additionally, they proposed a specialized construction for the two-server case with a communication complexity of $O(d^{\frac{1}{3}})$. Shortly after, Ambainis [1] significantly improved the upper bound for the n-server setting, reducing the complexity to $O(d^{\frac{1}{2n-1}})$. Beimel and Ishai [2] made further progress, considering a scenario in which up to t servers may collude. They presented a construction achieving a communication complexity of $O(d^{\frac{1}{\lfloor (2n-1)/t \rfloor}})$.

Wehner and de Wolf [11] made a key contribution to lower bounds in information-theoretic PIR, where they additionally parametrized the communication complexity by probe complexity b – a measure of the number of actual bits the client reads from the messages sent by the servers – as an additional parameter in the communication complexity analysis. In particular, they established a lower bound of $\Omega(d^{\frac{1}{b+1}})$ for the two-server case. Complementing this, they presented a scheme with $b = 1$ and communication complexity $O(\sqrt{d})$. Their results also demonstrated that the original scheme by Chor *et al.* [4], which with some modifications can be adapted to have $b = 3$, already achieved near-optimal complexity of $O(d^{\frac{1}{4}})$. Moreover, in the general case where the probe complexity equals the length of the messages sent by the two servers, their result implies a lower bound of $5\log d$, which remains the best-known lower bound for two servers.

For the two-server setting, the upper bound of Chor *et al.* was finally improved by Dvir and Gopi [5], who established an upper bound of $n^{O(\sqrt{\frac{\log\log d}{\log d}})}$, which remains the best known result for this case. In the multi-server setting, the currently best-known upper

[1] Note that there is a qualitative gap between our lower and upper bounds as they are currently presented: Our lower bound relies on perfect correctness, while our upper bound allows a failure to occur with some probability. The lower bound can be extended to account for a fixed probability of failure.

bound is due to works [6, 9, 3]. Efremenko [6] provided a three-server construction with communication complexity $exp(O(\sqrt{\log d \log\log d}))$ and a general 2^r-server construction with a communication complexity of $exp(O(\sqrt[r]{\log d \log^{r-1} \log d}))$. Recently, Ghasemi, Kopparty and Sudan [8] improved Efremenko's three-server construction, reducing the exponent from square root to the cube root. Other improvements – such as a reduction of the number of required servers of Efremenko's 2^r-server construction – were made by Itoh and Suzuki [9], and Chee *et al.* [3].

1.1 Technical Overview

In Section 2 we describe our definitions of non-interactive multi-server RPIR, adapted from Gentry *et al.* [7]. In Section 3 we describe our lower bound for the communication complexity of such protocols. We use the fact that in non-interactive two-server RPIR without correlated randomness, the client can combine any message $\mathtt{msg}_1$ from the first server with any message $\mathtt{msg}_2$ from the second server to retrieve a database row. So, if we take $\sqrt{d}$ messages $\mathtt{msg}_1$ and $\sqrt{d}$ messages $\mathtt{msg}_2$, the client can retrieve $\sqrt{d} \times \sqrt{d} = d$ database rows. Some of these may be the same, but we prove that, in expectation, the client ends up with cd *different* database rows for some constant c, which amounts to cdw bits of information. Communicating cdw bits of information naturally requires that many bits; so, if this much information can be extracted from $2\sqrt{d}$ messages, on average each of these messages must have size at least $\frac{cdw}{2\sqrt{d}} = O(\sqrt{d})w$.

In Section 4 we describe our sublinear upper bound (without correlated randomness). We build on the SimpleMSRPIR construction of Gentry *et al.*. In SimpleMSRPIR, one server sends a random database row, and the other server randomly pairs the rows and sends an XOR of each pair (as well as a concise description of the pairing). The client uses row i sent by the first server and the XOR of row i and row j sent by the second server to recover row j as well. With probability $\frac{1}{d}$ it returns row i; the rest of the time, it returns row j. In SimpleMSRPIR, the servers send a total of $\frac{dw}{2} + 2\log(d) + w$ bits, which is linear in the size dw of the database. In our construction (which we dub "Bucket-PIR"), instead of sending a single row, the first server sends each row with probability p. Instead of pairing the database rows, the second server partitions them into buckets of size b. By setting $p = \frac{1}{d^{\frac{\log\log(d)}{\log(d)}}}$ and $b = \frac{\log(d)}{\log\log(d)} + 1$, we ensure that the client retrieves a database row with probability at least half, while pushing the total communication complexity to be sublinear in the size dw of the database as long as $w = O(\log(d))$. (The probability that the client retrieves a row can be boosted arbitrarily close to 1 via parallel repetition; notice that the number of repetitions depends only on the desired probability of success, and not on the database size, so it does not impact the sublinearity of our communication complexity.)

In Section 5, we describe our final construction, which uses correlated randomness. We consider a setting with $n \geq 2$ servers, each of which holds Shamir shares of u random one-hot vectors (with zeros in all but one place) of length s. By locally taking the tensor product of all of its vectors of shares, each server ends up with shares of a size-d random one-hot vector, with a one at i. Each server takes the dot product of this vector of shares with the database, and sends the resulting value to the client. The client can treat the messages it receives as shares which reconstruct exactly to the ith database row.

2 Definitions

We use a definition of multi-server random-index PIR based on that of Gentry *et al.* [7, Definition 3]. Consider n servers $\mathcal{S}_1, \ldots, \mathcal{S}_n$, and a client $\mathcal{C}$. Since we limit ourselves to *non-interactive (one-round)* protocols, we can describe the entire protocol in terms of algorithms $\texttt{Server}_p$ (for $p \in [n]$) and $\texttt{Client}$. Each server $\mathcal{S}_p$ runs the algorithm $\texttt{Server}_p$ on the database D and randomness ρ_p (from distribution $\mathsf{R}_{\mathcal{S}}$) to obtain message $\texttt{msg}_p$. The client $\mathcal{C}$ runs the algorithm $\texttt{Client}$ on messages $\texttt{msg}_1, \ldots, \texttt{msg}_n$ and randomness $\rho_{\mathcal{C}}$ (from distribution $\mathsf{R}_{\mathcal{C}}$) to obtain (i, D_i).

Sometimes, the servers use database-independent correlated randomness. We use cr_p to denote server $\mathcal{S}_p$'s correlated randomness, and CR to denote the distribution from which $(\mathsf{cr}_1, \ldots, \mathsf{cr}_n)$ is drawn. We use grey to denote elements of an MS-RPIR scheme that are only present when correlated randomness is used.

▶ **Definition 1** (MS-RPIR: Multi-Server Random-Index PIR for Databases With d Rows of Size w). *A semi-honest, non-interactive (one-round)* c-correct n-server threshold-t random-index PIR scheme $(\{\texttt{Server}_p\}_{p\in[n]}, \texttt{Client}, \mathsf{CR})$ *must satisfy the following properties:*

c-Correctness ($0 < c \leq 1$): *For every database $D \in \{0,1\}^{w\times d}$ and index $i \in [d]$, the client's probability of outputting D_i is $\frac{c}{d}$.*
More formally, for every database $D \in \{0,1\}^{w\times d}$,

$$\Pr\left[\begin{array}{c} (i, D[i]) = \texttt{Client}(\texttt{msg}_1, \ldots, \texttt{msg}_n; \rho_{\mathcal{C}}) \\ \textit{where} \\ \rho_p \leftarrow \mathsf{R}_{\mathcal{S}} \textit{ for } p \in [n], \rho_{\mathcal{C}} \leftarrow \mathsf{R}_{\mathcal{C}}; \\ (\mathsf{cr}_1, \ldots, \mathsf{cr}_n) \leftarrow \mathsf{CR}; \\ msg_p \leftarrow \texttt{Server}(D, \mathsf{cr}_p; \rho_p) \textit{ for } p \in [n] \end{array}\right] = \frac{c}{d},$$

for every index $i \in [d]$. (Any other output of $\texttt{Client}$ *is interpreted to be $\perp$.)*

Client Privacy *For every database $D \in \{0,1\}^{w\times d}$, for every set of corrupt servers $I \subset [n]$ s.t. $|I| \leq t$, the index of the row output by the client is independent of the view of the corrupt servers.*
More formally, the following two distributions should be statistically indistinguishable:

$$\left\{\begin{array}{c} (i, \{\rho_p, \mathsf{cr}_p\}_{p\in I}) \\ \textit{where} \\ \rho_p \leftarrow \mathsf{R}_{\mathcal{S}} \textit{ for } p \in [n], \rho_{\mathcal{C}} \leftarrow \mathsf{R}_{\mathcal{C}}; \\ (\mathsf{cr}_1, \ldots, \mathsf{cr}_n) \leftarrow \mathsf{CR}; \\ msg_p \leftarrow \texttt{Server}(D, \mathsf{cr}_p; \rho_p) \textit{ for } p \in [n]; \\ (i, D[i]) = \texttt{Client}(\texttt{msg}_1, \ldots, \texttt{msg}_n; \rho_{\mathcal{C}}) \end{array}\right\} \sim \left\{\begin{array}{c} (i, \{\rho_p, \mathsf{cr}_p\}_{p\in I}) \\ \textit{where} \\ \rho_p \leftarrow \mathsf{R}_{\mathcal{S}} \textit{ for } p \in [n]; \\ (\mathsf{cr}_1, \ldots, \mathsf{cr}_n) \leftarrow \mathsf{CR}; \\ i \leftarrow [d] \textit{ with pr. } c, \ i = \perp \textit{ otherwise} \end{array}\right\}$$

▶ Remark 2. Notice that we allow the client to output $\perp$ with constant probability. Parallel executions can bring c down arbitrarily close to 0.

Finally, the trivial solution of the server(s) sending the client the entire database, and the client selecting a random index i and outputting $(i, D[i])$, satisfies the above definitions. In order to be interesting, a MS-RPIR scheme must also be *non-trivial*:

▶ **Definition 3** (Non-Triviality). *An MS-RPIR scheme is* non-trivial *if the communication complexity of the protocol is asymptotically smaller than the size of the database.*

More formally, let $\texttt{CC}(n, w, d)$ be the expected communication complexity of the protocol (that is, the expected sum of $|\texttt{msg}_1|, \ldots, |\texttt{msg}_n|$). The scheme is non-trivial if there exists a d_0 s.t. for all $d > d_0$, there exists a w_0 s.t. for all $w > w_0$,

$$\texttt{CC}(n, w, d) < wd.$$

3 A $\Omega(\sqrt{d}w)$ Lower Bound on Communication Complexity

In this section, we focus on the scenario where we have two servers, one of which is corrupt. In this setting, the servers must send $\Omega(\sqrt{d}w)$ bits. We will prove this by showing that choosing a random size $\sqrt{d}$ subset from the space of each server's possible messages in expectation allows recovering a constant fraction of the database.

▶ **Theorem 4** (1-out-of-2 IT MS-RPIR Lower Bound). *Any MS-RPIR protocol with perfect security for two servers, where one may be corrupt, must have at least $\Omega(\sqrt{d}w)$ expected communication.*

Proof. Let $D \in (\{0,1\}^w)^d$ be the database held by the two servers. For $i \in \{1,2\}$ let s_i be the number of possible random tapes for $\mathcal{S}_i$ and s_0 be the number of possible random tapes for $\mathcal{C}$. For a fixed random tape a server will always send the same message; some distinct tapes may result in the same message.

For servers $\mathcal{S}_1$ and $\mathcal{S}_2$ we define a matrix $M \in [d]^{s_1 \times s_2}$, indexed by the choices of $\mathcal{S}_1$ and $\mathcal{S}_2$ randomness. Entry $M_{r,c}$ encodes the index $i \in [d]$ which may be recovered by the client for server randomnesses r and c. More formally,

$$(i, D[i]) \leftarrow \texttt{Client}(\texttt{Server}_1(D;r), \texttt{Server}_2(D;c); T_{r,c}),$$

where client randomness $T_{r,c}$ is chosen uniformly and independently for each of the $s_1 \cdot s_2$ entries of M.

First, we address the case where $\mathcal{S}_1$ has fewer than $\sqrt{d}$ possible choices of randomness. If we take all messages that might be sent by $\mathcal{S}_1$ and any single message sent by $\mathcal{S}_2$ it must be possible to recover the entire database. This follows by correctness and privacy, as the client must always be able to recover some row of the database given a pair of messages, and all rows must still be equally likely even when fixing the message of one server. Iterating over all $< \sqrt{d}$ of $\mathcal{S}_1$ messages and client randomness will allow the recovery of the database, and therefore the messages of $\mathcal{S}_1$ (together with any one message of $\mathcal{S}_2$) on average contains $\geq \sqrt{d}w$ bits of information. A symmetric argument can be made for $\mathcal{S}_2$ with fewer than $\sqrt{d}$ possible random tapes.

From now on we will only consider the case where $s_1, s_2 \geq \sqrt{d}$. We will analyse how much of the database is recovered given server messages for k_1 choices of $\mathcal{S}_1$ randomness and k_2 choices of $\mathcal{S}_2$ randomness. Sample k_i distinct choices of randomness for server $i \in \{1,2\}$ as

$$R = (R_1, \dots, R_{k_1}),\ C = (C_1, \dots, C_{k_2}).$$

For all rows $i \in [d]$ in the database, and $a \in [k_1], b \in [k_2]$, define the random variable:

$$M^i_{R_a,C_b} = \begin{cases} 1 & \text{if } (i, D[i]) = \texttt{Client}(\texttt{Server}_1(D;R_a), \texttt{Server}_2(D;C_b); T_{a,b}) \\ 0 & \text{otherwise} \end{cases}$$

We may count the occurrences of each row $i \in [d]$ across the sampled rows and columns,

$$A^i_{R,C} = \sum_{a\in[k_1], b\in[k_2]} M^i_{R_a,C_b}.$$

And define an indicator variable for whether row $i \in [d]$ is recovered for any pair,

$$B^i_{R,C} = \begin{cases} 0 & \text{if } A^i_{R,C} = 0 \\ 1 & \text{otherwise} \end{cases}$$

To prove Theorem 4, it suffices to show that

$$E\left[\sum_{i\in[d]} B^i_{R,C}\right] \approx \frac{d}{2},$$

since this expectation describes how many database rows can be retrieved in expectation from our $\Omega(\sqrt{d})$ messages.[2]

By linearity of expectation, $E\left[\sum_{i\in[d]} B^i_{R,C}\right] = \sum_{i\in[d]} E[B^i_{R,C}]$; so, we can focus on $E[B^i_{R,C}] = \Pr[A^i_{R,C} > 0]$ for a single i.
Going forward we suppress the superscript i, as we will only handle one row $i \in [d]$ at a time. By the second moment method,

$$\Pr[A_{R,C} > 0] \geq \frac{E[A_{R,C}]^2}{E[A^2_{R,C}]}.$$

We can tackle $E[A_{R,C}]$ and $E[A^2_{R,C}]$ separately.

Bounding the expectation of $A_{R,C}$

First, we handle the simple case for $E[A_{R,C}]$. By linearity of expectation,

$$E[A_{R,C}] = \sum_{a\in[k_1],b\in[k_2]} E\left[M_{R_a,C_b}\right] = \sum_{a\in[k_1],b\in[k_2]} \Pr\left[M_{R_a,C_b}\right].$$

For all $c \in [s_2]$, privacy implies $\Pr[M_{R_a,c}] = 1/d$. Therefore, $\Pr\left[M_{R_a,C_b}\right] = 1/d$, and it follows,

$$E[A_{R,C}] = \sum_{a\in[k_1],b\in[k_2]} \frac{1}{d} = \frac{k_1 \cdot k_2}{d}.$$

Bounding the expectation of $A^2_{R,C}$

Expanding the definition and by linearity of expectation,

$$E[(A_{R,C})^2] = E\left[\left(\sum_{a\in[k_1],b\in[k_2]} M_{R_a,C_b}\right)^2\right] = \sum_{\substack{a\in[k_1],b\in[k_2]\\ c\in[k_1],d\in[k_2]}} E\left[M_{R_a,C_b} \cdot M_{R_c,C_d}\right].$$

We wish to derive an upper bound for this expectation. The terms in the sum may be split into four disjoint cases, (1) when $a = c$ and $b = d$, (2) when $a = c$ but $b \neq d$, (3) when $a \neq c$ but $b = d$, and (4) when $a \neq c$ and $b \neq d$.
We start with case (1) where $a = c$ and $b = d$,

$$\sum_{\substack{a=c\in[k_1]\\ b=d\in[k_2]}} E\left[M_{R_a,C_b} \cdot M_{R_c,C_d}\right] = \sum_{\substack{a=c\in[k_1]\\ b=d\in[k_2]}} \Pr\left[M_{R_a,C_b}, M_{R_c,C_d}\right] = \sum_{\substack{a\in[k_1]\\ b\in[k_2]}} \Pr\left[M_{R_a,C_b}\right]$$
$$= \sum_{a\in[k_1],b\in[k_2]} \frac{1}{d} = \frac{k_1 \cdot k_2}{d}.$$

[2] It would suffice to show that $E\left[\sum_{i\in[d]} B^i_{R,C}\right] \approx cd$ for any constant c; however, the math works out for $c = \frac{1}{2}$.

Before we proceed to the next cases, it will be helpful to define some notation. Let s_0 be the number of choices of client randomness. For $r \in [s_1]$ and $c \in [s_2]$ let $p_{r,c}$ be the number of client randomness choices for which the client outputs $(i, D[i])$ for server messages $\texttt{Server}_1(D; r), \texttt{Server}_2(D; c)$.

For any fixed $c \in [s_2]$ privacy implies the message i must occur for a fraction $1/d$ of the client choices and server 1 randomness,

$$\sum_{r=1}^{s_1} p_{r,c} = \frac{s_1 \cdot s_0}{d}.$$

Similarly, for any fixed $r \in [s_1]$,

$$\sum_{c=1}^{s_2} p_{r,c} = \frac{s_2 \cdot s_0}{d}.$$

Let $t_1 = s_1 s_0/d$ and $t_2 = s_2 s_0/d$. It will be convenient to consider a random variable $p_{X,Y}$ which is equal to $p_{r,c}$ when $X = r$ and $Y = c$.
Now we may proceed to case (2) when $a = c$ but $b \neq d$,

$$E\left[M_{R_a,C_b} \cdot M_{R_a,C_d}\right] = \Pr\left[M_{R_a,C_b}, M_{R_a,C_d}\right].$$

For any outcome $r \in [s_1]$ such that $R_a = r$,

$$\Pr[M_{r,C_b}, M_{r,C_d}] = \sum_{u \neq v \in [s_2]} \frac{p_{r,u}}{s_0} \cdot \frac{p_{r,v}}{s_0} \cdot \Pr[C_b = u, C_d = v] = \frac{1}{s_0^2 \cdot s_2(s_2-1)} \sum_{u \neq v \in [s_2]} p_{r,u} \cdot p_{r,v}$$

The sum may be simplified as,

$$\sum_{u \neq v \in [s_2]} p_{r,u} \cdot p_{r,v} = \left(\sum_{u \in [s_2]} p_{r,u}\right)^2 - \sum_{u \in [s_2]} p_{r,u}^2 = t_2^2 - \sum_{u \in [s_2]} p_{r,u}^2.$$

As $f(x) = x^2$ is convex we may apply Jensen's inequality to see $t_2^2/s_2 \leq \sum_{u \in [s_2]} p_{r,u}^2$,

$$\left(\frac{t_2}{s_2}\right)^2 = \left(\frac{\sum_{u=1}^{s_2} p_{r,u}}{s_2}\right)^2 \leq \frac{\sum_{u=1}^{s_2} (p_{r,u})^2}{s_2}.$$

Putting things together we see,

$$\begin{aligned} \Pr[M_{r,C_b}, M_{r,C_d}] &\leq \frac{1}{s_0^2 \cdot s_2(s_2-1)} \left(t_2^2 - \frac{t_2^2}{s_2}\right) = \frac{t_2^2}{s_0^2 \cdot s_2(s_2-1)} \left(1 - \frac{1}{s_2}\right) \\ &= \frac{s_0^2 \cdot s_2^2}{d^2 s_0^2 \cdot s_2(s_2-1)} \left(\frac{s_2 - 1}{s_2}\right) = \frac{1}{d^2}. \end{aligned}$$

For this case we may conclude, $E\left[M_{R_a,C_b} \cdot M_{R_a,C_d}\right] \leq \frac{1}{d^2}$.

Moving on to case (3) where $a \neq c$ but $b = d$, here an analogous argument to that of case (2) gives

$$E\left[M_{R_a,C_b} \cdot M_{R_c,C_b}\right] \leq \frac{1}{d^2}.$$

We proceed to the final case: (4) $a \neq c$ but $b \neq d$.

$$\begin{aligned} & E\left[M_{R_a,C_b} \cdot M_{R_c,C_d}\right] \\ & = \sum_{u\in[s_1]} \sum_{\substack{v\in[s_1]\\ u\neq v}} \sum_{n\in[s_2]} \sum_{\substack{m\in[s_2]\\ n\neq m}} \frac{p_{u,n}p_{v,m}}{s_0^2} \cdot \Pr[R_a = u, R_c = v, C_b = n, C_d = m] \\ & = \frac{1}{s_1(s_1-1)s_2(s_2-1)} \sum_{u\in[s_1]} \sum_{\substack{v\in[s_1]\\ u\neq v}} \sum_{n\in[s_2]} \sum_{\substack{m\in[s_2]\\ n\neq m}} \frac{p_{u,n}p_{v,m}}{s_0^2} \end{aligned}$$

Using Jensen's inequality as we did previously, for any $u \neq v \in [s_1]$,

$$\sum_{n\in[s_2]} \sum_{\substack{m\in[s_2]\\ n\neq m}} p_{u,n}p_{v,m} = \left(\sum_{n\in[s_2]} p_{u,n}\right)^2 - \sum_{n\in[s]} p_{u,n}^2 \leq t_2^2\left(\frac{s_2-1}{s_2}\right).$$

Therefore,

$$\begin{aligned} E\left[M_{R_a,C_b} \cdot M_{R_c,C_d}\right] & \leq \frac{1}{s_1(s_1-1)s_2(s_2-1)} \sum_{u\in[s_1]} \sum_{\substack{v\in[s_1]\\ u\neq v}} t_2^2\left(\frac{s_2-1}{s_2 s_0^2}\right) \\ & = \frac{s_1(s_1-1)(s_2-1)}{s_1(s_1-1)s_2^2(s_2-1)s_0^2} \cdot t_2^2 = \frac{1}{s_2^2 s_0^2} \cdot \left(\frac{s_2 s_0}{d}\right)^2 = \frac{1}{d^2}. \end{aligned}$$

Having bounded all terms for cases (2), (3) and (4) by $1/d^2$, we see

$$\sum_{\substack{a,c\in[k_1]\\ b,d\in[k_2]\\ a\neq c\vee b\neq d}} E\left[M_{R_a,C_b} \cdot M_{R_c,C_b}\right] \leq \frac{k_2k_1(k_1-1) + k_1k_2(k_2-1) + k_1(k_1-1)k_2(k_2-1)}{d^2},$$

letting $k_1 = k_2 = \sqrt{d}$ this simplifies to

$$\sum_{\substack{a,c\in[k_1]\\ b,d\in[k_2]\\ a\neq c\vee b\neq d}} E\left[M_{R_a,C_b} \cdot M_{R_c,C_b}\right] \leq \frac{2(\sqrt{d}-1) + (\sqrt{d}-1)^2}{d} = \frac{d-1}{d}.$$

Combining this with our bound from case (1) we arrive at

$$\sum_{\substack{a,c\in[\sqrt{d}]\\ b,d\in[\sqrt{d}]}} E\left[M_{R_a,C_b} \cdot M_{R_c,C_b}\right] \leq \frac{\sqrt{d}^2}{d} + \frac{d-1}{d} = 2 - \frac{1}{d},$$

which approaches 2 for increasing d.
So, we get

$$E[B_{R,C}^i] = \Pr[A_{R,C}^i > 0] \geq \frac{1}{2-\frac{1}{d}},$$

$$E\left[\sum_{i\in[d]} B_{R,C}^i\right] \geq \frac{d}{2-\frac{1}{d}} \approx \frac{d}{2},$$

as desired. ◀

SimpleMSRPIR [7]

`Server`$_1(D)$ **:** Pick $i \leftarrow [d]$ and return $\mathtt{msg}_1 = (i, D[i])$.

`Server`$_2(D)$ **:** Choose a random mask $\delta \leftarrow [d]$. If $\delta = 0$, set $D' = \perp$. otherwise, let $p_1, \dots, p_{d/2}$ be the list of pairs of indices $p_k = (j_{k,1}, j_{k,2})$ such that $j_{k,1} \oplus j_{k,2} = \delta$ (ordered by increasing smallest value in the pair). Define D' such that $D'[k] = D[j_{k,1}] \oplus D[j_{k,2}]$. ($D'$ has size $\frac{dw}{2}$.) Return $\mathtt{msg}_2 = (\delta, D')$.

`Client`$(\mathtt{msg}_1 = (i, D[i]), \mathtt{msg}_2 = (\delta, D'))$ **:** If $\delta = 0$, return $(i, D[i])$. Else, find p_k such that $i \in p_k$. Return $(i \oplus \delta, D'[k] \oplus D[i])$.

Figure 1 Description of SimpleMSPIR [7].

4 Bucket-RPIR

In this section, we present a construction based on splitting the database into random subsets, which we refer to as *buckets*. This construction is inspired by SimpleMSPIR of Gentry *et al.* [7], in which $\mathcal{S}_1$ sends a single random row, and $\mathcal{S}_2$ pairs the rows and sends the XOR of each pair, for total communication $\frac{dw}{2} + O(1)$. The client returns the row which $\mathcal{S}_1$'s row was paired with. We reproduce SimpleMSPIR in Figure 1.

SimpleMSPIR has communication complexity linear in the size of the database. In this section, we describe a generalization of SimpleMSPIR where, instead of pairing rows, $\mathcal{S}_2$ distributes the rows into larger buckets. In order to enable the client to retreive a row from XORs of larger sets, $\mathcal{S}_1$ sends many random rows instead of just one. Balancing the size of $\mathcal{S}_2$'s buckets and the number of rows $\mathcal{S}_1$ sends allows us to get communication complexity sublinear in the database size. We describe our protocol in Figure 2, where p is the probability that $\mathcal{S}_1$ sends any given row, and b is the size of $\mathcal{S}_2$'s buckets.

Bucket-RPIR

`Server`$_1(D)$ **:** Let $I = \emptyset$. For each row index $i \in [d]$, add i to I with probability p. Return $\mathtt{msg}_1 = \{(i, D[i])\}_{i \in I}$.

`Server`$_2(D)$ **:** Pick a random mapping $\pi : [d] \to [\frac{d}{b}]$ subject to the constraint that exactly b indices map to each value. (π can be represented in $d \log(\frac{d}{b})$ bits.) Define D' such that $D'[k] = D[i_1] \oplus \cdots \oplus D[i_b]$ where $\pi[i_1] = \cdots = \pi[i_b] = k$. ($D'$ has size $\frac{dw}{b}$.) Let $\mathtt{msg}_2 = (\pi, D')$.

`Client`$(\mathtt{msg}_1 = \{(i, D[i])\}_{i \in I}, \mathtt{msg}_2 = (\pi, D'))$ **:** Let π_k denote the set of indices $i_1, \dots, i_b$ such that $\pi[i_1] = \cdots = \pi[i_b] = k$.

If there does not exist a bucket k where exactly all but one of π_k are included in $\mathtt{msg}_1$, abort.

Otherwise, with probability $\frac{|\mathtt{msg}_1|}{d}$, output a random value from $\mathtt{msg}_1$.

With probability $1 - \frac{|\mathtt{msg}_1|}{d}$, pick a random bucket k for which all but one of π_k are included in $\mathtt{msg}_1$. Let i be such that $\pi(i) = k$ and i is missing from $\mathtt{msg}_1$. Return $(i, D'[k] \oplus (\oplus_{j \in \pi_k \setminus \{i\}} D[j]))$.

Figure 2 Description of Bucket-RPIR, Parametrized by p and b.

Next, we need to determine what p and b should be. The probability that the client is able to retrieve a row is

$$(\text{number of buckets}) \times \Pr[\text{all but one row in some bucket is sent by } \mathcal{S}_1]$$
$$= \frac{d}{b} \times b(1-p)p^{b-1} = d(1-p)p^{b-1}.$$

We need to pick b and p such that this probability is greater than half. Set

$$b = \frac{\log(d)}{\log\log(d)} + 1, p = \frac{1}{d^{\frac{\log\log(d)}{\log(d)}}}.$$

Let's check that with these parameters, the probability of success is indeed greater than half:

$$\Pr[\text{successful row retrieval}] = d(1-p)p^{b-1}$$
$$= d(1-p)\Big(\frac{1}{d^{\frac{\log\log(d)}{\log(d)}}}\Big)^{\frac{\log(d)}{\log\log(d)}+1-1} = 1-p.$$

For large enough d, p becomes smaller than half, and thus $1-p$ becomes greater than half.

Circling back to what each server sends:

$\mathcal{S}_1$ sends $pd = \frac{d}{d^{\frac{\log\log(d)}{\log(d)}}} = o(d)$ separate rows, which amounts to $o(d)w$ bits.

$\mathcal{S}_2$ sends $\frac{d}{b} = \frac{d}{\frac{\log(d)}{\log\log(d)}+1}$ XORed buckets, as well as a description of the function π. This amounts to $o(d)w + d\log(\frac{d}{b})$ bits.

The total amount of communicated information is thus sublinear in dw.

▶ **Theorem 5.** *Bucket-RPIR (Figure 2) is a secure MS-RPIR protocol (Definition 1) when p and b have the values described above.*

Proof. We start with correctness. The probability that the client outputs $\perp$ is equal to the probability that there is no bucket k such that $\texttt{msg}_1$ contains all but one row from π_k. We have set this probability to be $1-p$. Next, subject to the constraint that there is such a bucket, because of the randomness of π every row i not in $\texttt{msg}_1$ is equally likely to be the row missing from bucket k. The client chooses a row in $\texttt{msg}_1$ and not in $\texttt{msg}_1$ with proportional probabilities. So, Bucket-RPIR is $(1-p)$ correct.

We argue privacy against $\mathcal{S}_1$ and $\mathcal{S}_2$ separately. $\mathcal{S}_1$ learns nothing about i because, since the client chooses a row in $\texttt{msg}_1$ and not in $\texttt{msg}_1$ with proportional probabilities, whether a row is included or excluded from $\texttt{msg}_1$ does not correlate with its probability of being chosen. If a row in $\texttt{msg}_1$ is chosen, it is chosen randomly from $\texttt{msg}_1$; if a row outside of $\texttt{msg}_1$ is chosen, that choice is dictated by the choice of π, where any row outside of $\texttt{msg}_1$ is equally likely to be grouped with $b-1$ rows in $\texttt{msg}_1$.

$\mathcal{S}_2$ similarly learns nothing about i. This is because each row is included in $\texttt{msg}_1$ with the same probability, and due to the fact that π partitions the rows into buckets of equal size, each bucket is equally likely to be almost full (and each row is equally likely to be excluded from the almost-full bucket). ◀

▶ **Theorem 6.** *Bucket-RPIR (Figure 2) is non-trivial (Definition 3) when p and b have the values described above.*

Proof. $\mathcal{S}_1$ communicates $pd(\log(d)+w)$ bits; when $p = \frac{1}{d^{\frac{\log\log(d)}{\log(d)}}}$ and $w = \Omega(\log d)$, this is sublinear in dw. $\mathcal{S}_2$ communicates $d\log(\frac{d}{b}) + \frac{dw}{b}$ bits; when $b = \frac{\log(d)}{\log\log(d)} + 1$, this is also sublinear in dw. ◀

5 OneHot-RPIR

In this section, we present a construction based on one-hot vectors. We start by recalling the Shamir secret sharing scheme upon which we provide a high-level overview of the protocol and then offer a formal description in Figure 3.

5.1 Building Block: Secret Sharing

▶ **Definition 7** (t-out-of-n secret sharing scheme). *A t-out-of-n secret sharing scheme is a pair of algorithms* ($\texttt{share}, \texttt{reconstruct}$)*, such that:*

- $(s_1, \dots, s_n) \xleftarrow{\$} \texttt{share}(s, n, t; \rho)$ *is a randomized algorithm that, on input a secret s, outputs a set of n shares $(s_1, \dots, s_n)$ using randomness ρ and fixes the threshold t.*
- $s \leftarrow \texttt{reconstruct}(s_{i_1}, \dots, s_{i_t})$ *is a deterministic algorithm that on input t shares $(s_{i_1}, \dots, s_{i_t})$, where $(i_1, ..., i_t) \subset [n]$, outputs the secret s.*

A secret sharing scheme must satisfy the following properties:

Correctness: $\forall s, \forall T = (i_1, ..., i_t) \subset [n], \forall \rho$ *we have:*

$$\Pr_{\texttt{share}(s,n,t;\rho)\to(s_1,\dots,s_n)}[\texttt{reconstruct}(s_{i_1}, \dots, s_{i_t}) = s] = 1.$$

Perfect Security: $\forall s, s', \forall I \subseteq \{1, \dots, n\}, \forall \rho$ *where $|I| < n$, the following distributions are indistinguishable:*

$$\{(s_i : i \in I) \mid (s_1, \dots, s_n) \leftarrow \texttt{share}(s, n, t; \rho)\}$$
$$\{(s'_i : i \in I) \mid (s'_1, \dots, s'_n) \leftarrow \texttt{share}(s', n, t; \rho)\}$$

We describe the Shamir secret sharing scheme [10], which we use to instantiate the construction from expendable correlated randomness.

Let $\mathbb{F}_q$ be a finite field, where q is chosen as the smallest prime greater than n; and let $s \in \mathbb{F}_q$ be the secret we want to share. We start by picking a uniformly random polynomial $p \in \mathbb{F}_q[X]$ of degree $t-1$, such that $p(0) = s$, and for $i \in [n]$ define the shares $s_i = p(i)$. For $i \in [n]$ we distribute $p(i)$ to the i-th party. Now, observe that given that p is a $t-1$ degree polynomial, any set of at least t shares uniquely determines p. Upon collecting this set of at least t shares, we can reconstruct via polynomial interpolation.

For the sake of completeness, we formally describe the Shamir secret sharing scheme using the $\texttt{share}$ and $\texttt{reconstruct}$ algorithms:

▶ **Definition 8** (Shamir secret sharing scheme [10]).

- $\texttt{share}(s, n, t; \rho)$*: Using randomness ρ, sample $a_1, ..., a_{t-1} \xleftarrow{\rho} \mathbb{F}_q$, build the polynomial $p(x)$ as $p(x) = s + \sum_{i=1}^{t-1} a_i x^i$ and construct the shares $(s_1, \dots, s_n)$ as $s_i \leftarrow p(i)$.*
- $\texttt{reconstruct}(s_{i_1}, \dots, s_{i_t})$*: Interpolate the polynomial $p(x)$ for each $(i_1, ..., i_t) \subset [n]$ points corresponding to the secret shares and output $s = p(0)$.*

Properties

- **Additive homomorphism:** Let s^1 and s^2 be two secrets that we secret share using polynomials $p_1, p_2 \in \mathbb{F}_q$ both of degree $t-1$. Now, observe that in order to perform the addition of s^1 and s^2, each party can just locally add the two of its shares - e.g. the party P_i can compute $s_i^1 + s_i^2$ which corresponds to the i-th share of the polynomial $p_1 + p_2$ (of the same degree $t-1$) and therefore the secret $s^1 + s^2$.

- **Multiplicative homomorphism:** Let s^1 and s^2 be two secrets that we secret share using polynomials $p_1, p_2 \in \mathbb{F}_q$ of different degrees, $t_1 - 1$ and $t_2 - 1$. Now observe that if each party locally performs the multiplication of its shares, it obtains still a valid share of the polynomial $p_1 \cdot p_2$ but of degree $t_1 + t_2 - 2$. In order for the reconstruction to still naively work,[3] n must be such that $t_1 + t_2 - 1 \leq n$.

5.2 Overview of the construction

Let $\mathcal{S}_1, \ldots, \mathcal{S}_n$ denote the set of servers, where each server holds a copy of the database $D \in (\{0,1\}^w)^d$.

Let $\mathbf{v}_1, \ldots, \mathbf{v}_u$ represent a set of randomly generated one-hot vectors, each of length s, and let $\mathbf{v}_{i,j}$ denote the j-th component of the i-th one-hot vector. As the correlated randomness setup, we assume that each server holds a secret share of every component of every one-hot vector. More formally, server $\mathcal{S}_p$ holds a set of secret shares $\{\mathbf{v}^p_{i,j}\}_{i\in[u],j\in[s]}$, where $\mathbf{v}^p_{i,j}$ is the p-th share of the component $\mathbf{v}_{i,j}$, generated using the secret sharing algorithm $\texttt{share}(\mathbf{v}_{i,j}, n, t+1, \rho)$. We argue that although we heavily rely on this correlated randomness, this can easily be established in a pre-processing phase, for example, using a multi-party computation protocol.

Note that the tensor product of one-hot vectors results in another one-hot vector, but of a multiplied length (that is, the tensor product of our one-hot vectors will have length s^u). We refer to the resulting vector as the *tensored one-hot vector* $\mathbf{v}$. In the online phase, each server $\mathcal{S}_p$ computes the tensor product over its shares of the one-hot vectors. Specifically, each server computes

$$\mathbf{v}^p = \mathbf{v}^p_1 \otimes \cdots \otimes \mathbf{v}^p_u,$$

where for $i \in [u]$, the vector $\mathbf{v}^p_i$ is formed by stacking the shares $\{\mathbf{v}^p_{i,j}\}_{j\in[s]}$ into a single column vector.

Because Shamir sharing has a limited form of multiplicative homomorphism, by multiplying their shares locally as described above, the servers end up with shares of each component of the tensored one-hot vector. However, the degree of the sharing of the tensored one-hot vector will be the sum of the degrees of the sharings of the individual one-hot vectors. We must tune the threshold t (the degree of the polynomials used to share the individual components of the one-hot vectors) such that, after multiplication, there are still enough shares to reconstruct the vector. This can be achieved by setting the parameters t and u such that the number of servers $n > t \cdot u$.

For the final step of the protocol, each server computes the inner product of its shares of the tensored one-hot vector with the database. This corresponds to obtaining a share of the database row at the position where $\mathbf{v}^p$ is non-zero. Specifically, server $\mathcal{S}_p$ computes:

$$\texttt{msg}_p = \langle D, \mathbf{v}^p \rangle$$

and sends $\texttt{msg}_p$ to the client. To ensure correctness, we set the size of the original one-hot vectors s such that $s^u = d$, where d is the number of rows in the database.

Upon receiving the messages $\texttt{msg}_1, \ldots, \texttt{msg}_n$, the client obtains a random row of the database by computing:

$$D_* \leftarrow \texttt{reconstruct}(\texttt{msg}_1, \ldots, \texttt{msg}_n).$$

[3] To work without using techniques such as degree reduction that require additional communication.

MS-RPIR Protocol based on One-Hot Vectors

Servers $\mathcal{S}_1, ..., \mathcal{S}_n$ each hold the database $D \in (\{0,1\}^w)^d$.
Setup: Each server $\mathcal{S}_p$ for $p \in [n]$ holds shares of one-hot vectors $\mathbf{v}_1, \ldots, \mathbf{v}_u$ of length s. These shares are computed over the individual components of the vectors as $\texttt{share}(\mathbf{v}_{i,j}, n, t+1, \rho)$, for $i \in [u], j \in [s]$, where $\mathbf{v}_{i,j}$ is the j-th component of the vector $\mathbf{v}_i$. Namely, each server $\mathcal{S}_p$ holds $\{\mathbf{v}^p_{i,j}\}_{i\in[u],j\in[s]}$.

Online Phase:
$\texttt{Server}_p(D, \{\mathbf{v}^p_{i,j}\}_{i\in[u],j\in[s]})$: Compute $\mathbf{v}^p = \mathbf{v}^p_1 \otimes \cdots \otimes \mathbf{v}^p_u$, where for $i \in [u]$, the vector $\mathbf{v}^p_i$ is formed by stacking the shares $\{\mathbf{v}^p_{i,j}\}_{j\in[s]}$ into a single column vector. Set $\texttt{msg}_p = \langle D, \mathbf{v}^p \rangle$ and send $\texttt{msg}_p$ to the client.
$\texttt{Client}(\texttt{msg}_1, \ldots, \texttt{msg}_n)$: Compute $D_* \leftarrow \texttt{reconstruct}(\texttt{msg}_1, \ldots, \texttt{msg}_n)$ and output $(*, D_*)$.[a]

[a] We assume that the database has the index of each row encoded inside that row.

Figure 3 Description of the One-Hot Vectors Protocol.

▶ **Theorem 9.** *Let q be a $\lceil \max\{\log(d) + w, \log(n)\}\rceil$-bit prime and $\mathbb{F}_q$ be a finite field. Then the one-hot vectors protocol from Figure 3 is a non-trivial n-server threshold-t MS-RPIR protocol, satisfying Definition 1, with communication complexity* $\texttt{CC}(n, w, d) = n(\log(d)w + 1)$.

The protocol is *non-trivial* (Definition 3), since we replace communication complexity with pre-processing size. During the online phase, each server sends one message of size $\lceil \log q \rceil$; since the client obtains n messages (one from each server), the communication complexity is

$$\texttt{CC}(n, w, d) = n\lceil \log q \rceil = n \lceil \max\{\log(d) + w, \log(n)\}\rceil .$$

The protocol also requires pre-processing of size non-trivially less than the number of rows d as long as $u > 1$, which can only happen as long as $n > 2t$.

Proof.

Correctness. Let $f_{i,j}(x)$ be the random polynomial used for the secret sharing of the j-th component of the i-th one-hot vector. After establishing the setup, the server $\mathcal{S}_p$ has shares $\{f_{i,j}(p)\}_{i\in[u],j\in[s]} = \{\mathbf{v}^p_{i,j}\}_{i\in[u],j\in[s]}$ which can be rearranged as a set of one-hot vectors $\{\mathbf{v}^p_1, \mathbf{v}^p_2, ..., \mathbf{v}^p_u\}$, where

$$\mathbf{v}^p_i = \begin{bmatrix} \mathbf{v}^p_{i,1} \\ \mathbf{v}^p_{i,2} \\ \vdots \\ \mathbf{v}^p_{i,s} \end{bmatrix}. \tag{1}$$

Note, however, that only one of the rows corresponds to a share of 1, whereas the other rows correspond to some share of 0.

Next, each server $\mathcal{S}_p$ computes the tensor product $\mathbf{v}^p = \mathbf{v}^p_1 \otimes \mathbf{v}^p_2 \otimes ... \otimes \mathbf{v}^p_u$, which more closely looks like

$$\mathbf{v}^p = \begin{bmatrix} \mathbf{v}^p_{1,1} \cdot \mathbf{v}^p_{2,1} \cdots \mathbf{v}^p_{u-1,1} \cdot \mathbf{v}^p_{u,1} \\ \mathbf{v}^p_{1,1} \cdot \mathbf{v}^p_{2,1} \cdots \mathbf{v}^p_{u-1,1} \cdot \mathbf{v}^p_{u,2} \\ \vdots \\ \mathbf{v}^p_{1,1} \cdot \mathbf{v}^p_{2,1} \cdots \mathbf{v}^p_{u-1,1} \cdot \mathbf{v}^p_{u,s} \\ \\ \mathbf{v}^p_{1,1} \cdot \mathbf{v}^p_{2,1} \cdots \mathbf{v}^p_{u-1,2} \cdot \mathbf{v}^p_{u,1} \\ \mathbf{v}^p_{1,1} \cdot \mathbf{v}^p_{2,1} \cdots \mathbf{v}^p_{u-1,2} \cdot \mathbf{v}^p_{u,2} \\ \vdots \\ \mathbf{v}^p_{1,1} \cdot \mathbf{v}^p_{2,1} \cdots \mathbf{v}^p_{u-1,2} \cdot \mathbf{v}^p_{u,s} \\ \vdots \\ \vdots \\ \vdots \\ \mathbf{v}^p_{1,s} \cdot \mathbf{v}^p_{2,s} \cdots \mathbf{v}^p_{u-1,s} \cdot \mathbf{v}^p_{u,1} \\ \mathbf{v}^p_{1,s} \cdot \mathbf{v}^p_{2,s} \cdots \mathbf{v}^p_{u-1,s} \cdot \mathbf{v}^p_{u,2} \\ \vdots \\ \mathbf{v}^p_{1,s} \cdot \mathbf{v}^p_{2,s} \cdots \mathbf{v}^p_{u-1,s} \cdot \mathbf{v}^p_{u,s} \end{bmatrix} \tag{2}$$

Lastly, the server computes the message $\mathtt{msg}_p$ as the inner product $\langle \mathbf{v}^p, D\rangle$. Let the indices of the database range from 0 to $d-1$. Then, we can rewrite

$$\mathtt{msg}_p = \langle D, \mathbf{v}^p \rangle = \sum_{s_1,\ldots,s_u \in [s]} D[\sum_{i\in[u]} (s_i - 1) \cdot s^{u-i}] \prod_{j\in[u]} \mathbf{v}^p_{j,s_j}.$$

Notice that this expression is a summation of s^u values where each of the values is a database row (some constant in $\mathbb{F}_t$), multiplied by a product of exactly u shares. Since every share corresponds to a polynomial of degree t, the product of u shares corresponds to a secret share of a polynomial of degree $t \cdot u$. As the client obtains $n > t \cdot u$ shares, it can successfully interpolate the polynomial in all n points.

Observe again that since $\mathbf{v}^p$ is a tensor product of u one-hot vectors, only one row is a share of 1 while the others are shares of 0. With that follows the correctness of the client's output.

The argument that any row in the database is obtained with the same probability $\frac{1}{d}$ follows directly from the randomness of one-hot vectors $\mathbf{v}_1, ..., \mathbf{v}_u$, as the position of the non-zero row in each of the vectors is picked uniformly at random results in the position of the non-zero row in the vector $\mathbf{v}$, obtained by the tensor product $\mathbf{v} = \mathbf{v}_1 \otimes ... \otimes \mathbf{v}_u$ also being uniformly random.

Client Privacy. To argue for client privacy, we need to show that the distribution of the index output by the client, along with the view of up to threshold-t colluding servers, is indistinguishable from the distribution where the index is sampled uniformly at random. This follows from the randomness of the one-hot vectors: the index of the row received by the client corresponds to the position of the non-zero entry in the one-hot vector $\mathbf{v}$, which is computed as the tensor product $\mathbf{v} = \mathbf{v}_1 \otimes \cdots \otimes \mathbf{v}_u$. As in the correctness argument, each vector $\mathbf{v}_1, \ldots, \mathbf{v}_u$ has its non-zero position chosen uniformly at random. Consequently, the non-zero position in $\mathbf{v}$ is also uniformly random, ensuring that the client's obtained index remains uniformly distributed.

The view of up to threshold-t colluding servers is exactly t shares of each vector $\mathbf{v}_1, \ldots, \mathbf{v}_u$. As we use the Shamir sharing scheme, which has perfect security, this set of shares reveals no information about the non-zero position in vectors and the index obtained by the client. Therefore, the two distributions are indistinguishable. ◀

References

1 Andris Ambainis. Upper bound on communication complexity of private information retrieval. In Pierpaolo Degano, Roberto Gorrieri, and Alberto Marchetti-Spaccamela, editors, *ICALP 97*, volume 1256 of *LNCS*, pages 401–407. Springer, Heidelberg, July 1997. `doi:10.1007/3-540-63165-8_196`.

2 Amos Beimel and Yuval Ishai. Information-theoretic private information retrieval: A unified construction. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *ICALP 2001*, volume 2076 of *LNCS*, pages 912–926. Springer, Heidelberg, July 2001. `doi:10.1007/3-540-48224-5_74`.

3 Yeow Meng Chee, Tao Feng, San Ling, Huaxiong Wang, and Liang Feng Zhang. Query-efficient locally decodable codes of subexponential length, 2010. `arXiv:1008.1617`.

4 Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th FOCS*, pages 41–50. IEEE Computer Society Press, October 1995. `doi:10.1109/SFCS.1995.492461`.

5 Zeev Dvir and Sivakanth Gopi. 2-server PIR with sub-polynomial communication. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 577–584. ACM Press, June 2015. `doi:10.1145/2746539.2746546`.

6 Klim Efremenko. 3-query locally decodable codes of subexponential length. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 39–44. ACM Press, May / June 2009. `doi:10.1145/1536414.1536422`.

7 Craig Gentry, Shai Halevi, Bernardo Magri, Jesper Buus Nielsen, and Sophia Yakoubov. Random-index PIR and applications. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part III*, volume 13044 of *LNCS*, pages 32–61. Springer, Heidelberg, November 2021. `doi:10.1007/978-3-030-90456-2_2`.

8 Fatemeh Ghasemi, Swastik Kopparty, and Madhu Sudan. Improved pir schemes using matching vectors and derivatives, 2024. `doi:10.48550/arXiv.2411.11611`.

9 Toshiya ITOH and Yasuhiro SUZUKI. Improved constructions for query-efficient locally decodable codes of subexponential length. *IEICE Transactions on Information and Systems*, E93-D(2):263–270, 2010. `doi:10.1587/transinf.e93.d.263`.

10 Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979. `doi:10.1145/359168.359176`.

11 Stephanie Wehner and Ronald de Wolf. Improved lower bounds for locally decodable codes and private information retrieval. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP 2005*, volume 3580 of *LNCS*, pages 1424–1436. Springer, Heidelberg, July 2005. `doi:10.1007/11523468_115`.