One Extra Bit of Download Ensures Perfectly Private Information Retrieval

Nihar B. Shah, K. V. Rashmi, Kannan Ramchandran, *Fellow, IEEE*

***Abstract*—Private information retrieval (PIR) systems allow a user to retrieve a record from a public database without revealing to the server which record is being retrieved. The literature on PIR considers only replication-based systems, wherein each storage node stores a copy of the *entire* data. However, systems based on erasure codes are gaining increasing popularity due to a variety of reasons. This paper initiates an investigation into PIR in erasure-coded systems by establishing its capacity and designing explicit codes and algorithms. The notion of privacy considered here is information-theoretic, and the metric optimized is the amount of data downloaded by the user during PIR.**

In this paper, we present four main results. First, we design an explicit erasure code and PIR algorithm that requires only *one extra bit* of download to provide perfect privacy. In contrast, all existing PIR algorithms require a download of at least twice the size of the requisite data. Second, we derive lower bounds proving the necessity of downloading at least one additional bit. This establishes the precise *capacity* of PIR with respect to the metric of download. These results are also applicable to PIR in replication-based systems, which are a special case of erasure codes. Our third contribution is a negative result showing that capacity-achieving codes necessitate super-linear storage overheads. This motivates the fourth contribution of this paper: an erasure code and PIR algorithm that requires a linear storage overhead, provides high reliability to the data, and is a small factor away from the capacity.

I. Introduction

Private information retrieval (PIR) systems allow a user to retrieve a record from a public database without revealing to the server (or the storage "node") which record is being retrieved. PIR finds application when users want to retrieve privacy-sensitive information like medical data, stock data, patent data, etc. [1]–[3] (see [4] for an excellent survey).

In this paper, we consider the information-theoretic (and not the computational) notion of PIR, where the node must not obtain any information about the record being retrieved even if the node possesses an infinite computational ability. A trivial means of performing PIR is for the user to download the whole database. This solution is clearly impractical due to the massive amount of download required, but is the only feasible solution if there is only a single storage node storing the data [5]. The amount of communication can however be reduced if there are multiple (non-colluding) nodes [4]-[9].

Existing algorithms for private information retrieval (e.g., [4]-[9]) assume the storage system to be *replication-based*, i.e., they assume the system to comprise multiple storage nodes, each of which stores a copy of the *entire database*. However, an alternative means of storing the data is gaining significant popularity today: that of using *erasure codes* [10]-[12].

Erasure codes encode and store data across multiple nodes, with each node requiring to store data that is only a fraction of the size of the original data. Erasure codes increase the reliability and availability of the data while significantly reducing the total storage requirements. As a result, erasure codes have become increasingly popular in distributed storage systems [10]-[12], which motivates us to investigate PIR under erasure codes. Another motivator for erasure-coded PIR systems is in providing provable privacy primitives in large-scale scenarios where nodes are distributed and have limited storage capacities, for example, to empower dissidents in the face of oppressive governments where no single node should store the data in its entirety [13].

To the best of our knowledge, the problem of PIR under erasure-coded nodes has not been investigated previously in the literature. This is what this paper initiates by providing capacity results as well as explicit algorithms for erasure-coded PIR systems. Replication is a special case of erasure codes, and hence our results also carry over to replication-based systems. In this paper, we focus on optimizing the metric of the *amount of download* that the user is required to perform when privately retrieving the data. We assume that the nodes are non-colluding, i.e., they cannot cooperate with each other to recover information about the query. Fig. 1 shows an example of our capacity-achieving erasure code and PIR algorithm.

The rest of this paper is organized as follows. Section II describes the problem setting and summarizes our results. Section III proves the necessity of downloading at least one additional bit for PIR in the worst case. This section also shows that achieving this lower bound necessitates super-linear storage overheads. Section IV then presents a simple explicit code that achieves this lower bound, thereby establishing the capacity of PIR with respect to the metric of download. Section V presents an erasure code and PIR algorithm with a linear storage overhead that is a small factor away from capacity. Finally, Section VI presents concluding remarks.

II. Problem Setting & Summary of Results

Consider a database comprising k records, each of which is of size R bits. There are n storage nodes in the network across which this database is to be (encoded and) stored. Each node can store upto α bits of data. A user may wish to recover any record $z \in [k]$ (for any positive integer A, $[A]$ denotes the set $\{1, \ldots, A\}$). A private information retrieval (PIR) algorithm is one which allows the user to retrieve the desired record by downloading data from (a subset of) the n nodes in a manner that no node can obtain any information regarding which record z is being retrieved. The goal is to design erasure

The authors are with the Dept. of EECS, UC Berkeley. Email: {nihar, rashmikv kannanr}@eecs.berkeley.edu. N. B. Shah was supported in part by a Berkeley fellowship and K. V. Rashmi by a Facebook fellowship and an MSR fellowship. This work was also supported in part by NSF CCF-0964018, NSF CCF-1116404, and MURI CHASE grant no. 556016.

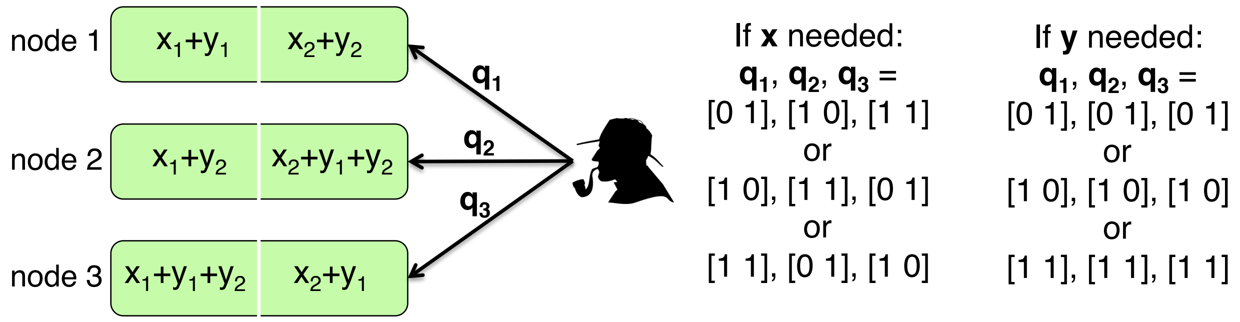

Fig. 1: An example of our capacity-achieving PIR algorithm and erasure code. Two records $\mathbf{x} = [x_1 \quad x_2]$ and $\mathbf{y} = [y_1 \quad y_2]$, each of size $R = 2$ bits, are stored across three (non-colluding) storage nodes. The code operates in the binary field, i.e., addition operations are XORs. One can verify that the code ensures no loss of data even if any one of the three nodes fail (and the code is *Maximum-Distance-Separable*). Now, the user may wish to recover either of the two records $\mathbf{x}$ or $\mathbf{y}$, but without revealing which of these is being recovered. For each $i \in \{1, 2, 3\}$, the user sends the 2-bit query $\mathbf{q}_i$ to node i. The choice of $(\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3)$ depends on the identity of the record to be retrieved, and the choice among the multiple options (denoted as "or" in the figure) is made uniformly at random. Each node i returns an inner-product of its data with the received query $\mathbf{q}_i$ (for instance, if $\mathbf{q}_1 = [1 \quad 1]$ then node 1 returns $(x_1 + y_1 + x_2 + y_2)$). One can verify that the user can obtain the desired record from the downloaded data. On the other hand, from the perspective of any node i, the likelihood of observing any value of $\mathbf{q}_i$ is identical across the two records, which prevents it from obtaining any information about the identity of the desired record. The total amount of data downloaded in the PIR operation is $(R + 1) = 3$ bits which is the minimum possible.

codes and PIR algorithms that *minimize* the amount of data that the user has to *download* when performing PIR.

In this paper, we consider the information-theoretic notion of PIR. We assume that the nodes do not collude. We also assume that nodes are passive eavesdroppers, i.e., they do not corrupt any data but may store and utilize any available information. The queries for any PIR operation are allowed to be static or adaptive, deterministic or random.

We define the *storage overhead* of the system as the ratio of the total storage used in the system to the total size of all the records:

$$\text{storage overhead} = \frac{n\alpha}{kR} \ .$$

We say that the storage overhead is *super-linear* if $\lim_{R\to\infty} \text{storage overhead} \rightarrow \infty$, and *linear* if $\lim_{R\to\infty} \text{storage overhead} \rightarrow c$ for some finite constant c. (Note that n and α may vary with R; k is fixed and independent of R.)

The problem of PIR in erasure codes has several metrics of possible interest such as the amount of download, query-size and connectivity required for PIR, the storage-space occupied and the reliability offered by the erasure code. While a grander goal would be to establish the tradeoffs and relationships among all of these metrics, this paper focuses on the metric of amount of *download* required for PIR. With the goal of establishing the *capacity* of PIR in terms of the download, we ask the following question: if no constraints are imposed, then what is the minimum possible worst-case download for PIR? The answer is proved to be $(R + 1)$ bits in this paper.

We show that any algorithm achieving the capacity must incur a storage that is super-linear in R and a connectivity (and hence number of nodes in the system) that is linear in R. Moreover, the underlying erasure code may not have good reliability properties. We then ask the question: if the storage capacity is constrained to be linear in R, the number of nodes to be independent of R, and if the erasure code must have a 'high' reliability, then what can be achieved? To this end, we design an explicit erasure code and PIR algorithm that satisfies these constraints, with each node storing an amount less than twice the size of a record and all the k records being recoverable from any k nodes (ensuring reliability in the presence of $(n - k)$ arbitrary node-failures), and the PIR algorithm entailing a download that is away from capacity by a factor less than 4.

The **extended version [14] of this paper on arXiv** contains additional results such as (a) capacity-characterization when storage nodes can collude, (b) PIR algorithms for Maximum-Distance-Separable (MDS) codes, (c) proof that achieving the capacity necessitates a super-linear storage, and (d) implications of these results on locally decodable codes.

III. Lower Bounds on Download for PIR

Theorem 1: Suppose there are $k \geq 3$ records, each of size R bits, stored using any arbitrary erasure code (including replication). Then (a) any PIR algorithm must download at least $(R + 1)$ bits in the worst case. Any PIR algorithm that downloads at most $(R + 1)$ bits in the worst case must also satisfy, for almost every PIR operation: (b) $(R + 1)$ bits must be downloaded, (c) the user must connect to at least $(R + 1)$ nodes. Furthermore, (d) the total amount of storage must be super-linear in R.

Proof: For any function f, let us denote the cardinality of its range as $\rho(f)$ $(\in \{1, 2, \ldots\})$. For any collection of functions $f_1, \ldots, f_h$, denote their Cartesian product as $f_1 \times \cdots \times f_h$.

We first prove part (a). To this end, consider any PIR algorithm that downloads *at most* $(R + 1)$ bits during PIR in the worst case. Consider any one record, and for convenience, term it the 'first' record. Now suppose the user contacts some storage node (let us call it the 'first' node) in a PIR operation for retrieving the first record. Suppose the PIR algorithm asks the first node to pass some function f of its data. Without loss of generality, assume that function f is not a constant function, i.e., $\rho(f) \geq 2$. Further, let g_1 denote the Cartesian product of the functions asked from all other nodes in this instance of PIR. Upon receiving the values of these functions, the user must be able to recover the first record. Since each record is of R bits, these functions must satisfy $2^R \leq \rho(f \times g_1) \leq \rho(f)\rho(g_1)$.

Since the download of function f from the first node and function g_1 from the remaining nodes cannot comprise more than $(R+1)$ bits in total, it must be that $\rho(f)\rho(g_1) \leq 2^{R+1}$. In order to ensure that the first node does not identify the record being recovered, there must also be instances of retrieval of records $z \in \{2, \ldots, k\}$ in which function f is queried from the first node. Let g_z be the Cartesian product of the functions downloaded from other nodes in such an instance of PIR of record $z \in \{2, \ldots, k\}$. From arguments similar to those made above for the first record, we get

$$2^{R+1} \geq \rho(f)\rho(g_z) \geq 2^R \quad \forall z \in [k] \ .$$

Multiplying the left hand sides of this equation across all $z \in [k]$, we get

$$2^{kR+k} \geq (\rho(f))^k \rho(g_1) \cdots \rho(g_k) \ . \tag{1}$$

Now, since the functions $f, g_1, \ldots, g_k$ together suffice to reconstruct all the k records, it must be that

$$2^{kR} \leq \rho(f \times g_1 \times \cdots \times g_z) \leq \rho(f)\rho(g_1) \cdots \rho(g_z) \ .$$

Along with (1) and our assumption of $k \geq 3$, this gives

$$\rho(f) \leq 2^{\frac{k}{k-1}} \leq 2^{\frac{3}{2}} < 2.83 \ .$$

Thus, $\rho(f) \in \{1, 2\}$, and since the function f is not a constant function, it must be that $\rho(f) = 2$. Since the choice of the 'first' node and the 'first' record was arbitrary, it follows that any (non-constant) function queried from any node for PIR must have a range with a cardinality exactly equal to 2. Note that so far in the proof, we have allowed a download of upto $(R+1)$ bits for any instance of PIR.

Now consider retrieval of the first record, and suppose the user queries some Δ nodes. Let (non-constant) functions $f_1, \ldots, f_\Delta$ be these respective queries. From the arguments above, we must have that $\rho(f_i) = 2 \ \forall \ i \in [\Delta]$. Thus we have

$$\rho(f_1)\rho(f_2) \cdots \rho(f_\Delta) = 2^\Delta \ . \tag{2}$$

If $\Delta < R$ then $\rho(f_1)\rho(f_2) \cdots \rho(f_\Delta) < 2^R$, making it impossible to guarantee retrieval of the record of R bits from these functions. Suppose $\Delta = R$. Then the amount of download is precisely equal to the size of the required record, and hence each of the functions f_i must be functions of only the first record (and must be independent of the values of all other records). If such an event has a high likelihood under the (possibly randomized) PIR algorithm, then it provides the nodes non-zero information about the identity of the required record: in this situation, if the nodes adopt a protocol of identification as "if the function queried is a function of only one record, then output that record as the required record, else output a random record as the desired record," then the nodes will identify the required record correctly strictly more than $\frac{1}{k}$ of the time, i.e., strictly better than random guessing. It follows that almost always we must have $\Delta \geq (R+1)$, i.e., the user must connect to $(R+1)$ or more nodes and download at least one bit from each of them. This amounts to a total download of $(R+1)$ or higher. This last argument also proves parts (b) and (c).

The proof of (d) is available in the extended version [14] of this paper on arXiv. ■

Note that while Theorem 1 considers $k \geq 3$, the lower bound of $(R+1)$ bits also holds for $k = 2, R = 2$ considered in Fig. 1.

IV. Capacity Achieving Code and PIR Algorithm

In this section, we present a simple explicit code and PIR algorithm that achieves the lower bound of $(R+1)$ bits on the download, thus establishing the capacity of PIR as $(R+1)$ with respect to the download. Fig. 1 depicts an example of our capacity-achieving construction.

Theorem 1 mandates any capacity-achieving PIR algorithm to have a storage that is super-linear in R; the storage required by the code presented here is polynomial in R. The code is linear and operates over the binary field (additions and subtractions are XOR operations). The description below assumes R to be an even number (the case when it is odd requires a small modification and is described subsequently).

Encoding: We first introduce some notation. Let $\mathbf{1}$ denote the all-ones column of length $(R+1)$. For $i \in [R]$, let $\mathbf{u}_i$ denote the i^{th} unit vector of length R. All vectors are column vectors by default. Let $\mathbf{u}_{R+1} := \sum_{i=1}^{R} \mathbf{u}_i = \mathbf{1}$. Define an $((R+1) \times R)$ matrix U as

$$U := [\mathbf{u}_1 \quad \cdots \quad \mathbf{u}_{R+1}]^T.$$

Let $\mathcal{P}$ denote the set of the $(R+1)$ cyclic permutation matrices of size $((R+1) \times (R+1))$. Let $\mathcal{P}^{k-1}$ denote the $(k-1)$-fold Cartesian product of $\mathcal{P}$ with itself. Observe that the set $\mathcal{P}^{k-1}$ has $(R+1)^{k-1}$ elements. Denote these $(R+1)^{k-1}$ elements as $(P_2^{(i)}, \cdots, P_k^{(i)})$ for $i \in [(R+1)^{k-1}]$, where each $P_j^{(i)} \in \mathcal{P}$ is a cyclic permutation matrix of size $((R+1) \times (R+1))$.

Consider a system with $n = (R+1)^{k-1}$ nodes, each having a storage space of $\alpha = R$ bits. For $i \in [k]$, define $\mathbf{m}_i$ to be a R-length binary vector comprising the i^{th} record. Let $\mathbf{m}$ be a binary vector of length kR comprising all the records $\mathbf{m}^T = [\mathbf{m}_1^T \quad \cdots \quad \mathbf{m}_k^T]$. Algorithm 1 now describes the encoding procedure.

Algorithm 1 Encoding

```
For $i \in [(R+1)^{k-1}]$, define an $(R+1)$-length vector $\mathbf{c}^{(i)}$ as
    $\mathbf{c}^{(i)} := [U \quad P_2^{(i)}U \quad P_3^{(i)}U \quad \cdots \quad P_k^{(i)}U] \mathbf{m}$ .
Node $i$ stores the first $R$ bits of this vector $\mathbf{c}^{(i)}$.
```

Observe that $\mathbf{1}^T U$ is an all-zero vector. As a result,

$$\begin{aligned} \mathbf{1}^T \mathbf{c}^{(i)} &= \left[\mathbf{1}^T U \quad \mathbf{1}^T P_2^{(i)} U \quad \cdots \quad \mathbf{1}^T P_k^{(i)} U\right] \mathbf{m} \\ &= \left[\mathbf{1}^T U \quad \mathbf{1}^T U \quad \cdots \quad \mathbf{1}^T U\right] \mathbf{m} \qquad (3) \\ &= 0 \qquad (4) \end{aligned}$$

where (3) is because the sum of all the rows of a cyclic permutation matrix is an all ones vector. Now, (4) implies that the $(R+1)^{\text{th}}$ bit of $\mathbf{c}^{(i)}$ is simply an XOR of the first R bits. Thus, for the purposes of the PIR algorithm for retrieving data, we can assume that each node i equivalently "stores" the entire vector $\mathbf{c}^{(i)}$.

The example of Fig. 1 has $n = 3$, $R = 2$, $k = 2$, $\alpha = 2$, $\mathbf{m} = [x_1\ x_2\ y_1\ y_2]^T$, $U = [1\ 0; 0\ 1; 1\ 1]$, $P_2^{(1)} = [1\ 0\ 0; 0\ 1\ 0; 0\ 0\ 1]$, $P_2^{(2)} = [0\ 1\ 0; 0\ 0\ 1; 1\ 0\ 0]$, $P_2^{(3)} = [0\ 0\ 1; 1\ 0\ 0; 0\ 1\ 0]$.

PIR Algorithm: Algorithm 2 describes our capacity-achieving PIR algorithm. Note that the notation of Algorithm 2 differs from Fig. 1 in that Algorithm 2 directly asks for a specific bit, whereas Fig. 1 depicts an equivalent method of asking a linear combination.

Algorithm 2 Capacity-achieving PIR Algorithm

```
***Query construction:***
Let $z\in[k]$ denote the desired record.
Let $\{\mathbf{u}_1,\ldots,\mathbf{u}_{R+1}\}^{k-1}$ denote the $(k-1)$-fold Cartesian
    product of the set $\{\mathbf{u}_1,\ldots,\mathbf{u}_{R+1}\}$ with itself.
Choose the set of vectors $\{\mathbf{v}_1,\ldots,\mathbf{v}_{z-1},\mathbf{v}_{z+1},\ldots,\mathbf{v}_k\}$ uni-
    formly at random from $\{\mathbf{u}_1,\ldots,\mathbf{u}_{R+1}\}^{k-1}$ (each $\mathbf{v}_i$ is of
    length $R$).
Connect to $(R+1)$ of the nodes that respectively store the
    $(R+1)$ bits $[\mathbf{v}_1^T\ \cdots\ \mathbf{v}_{z-1}^T\ \mathbf{u}_j^T\ \mathbf{v}_{z+1}^T\ \cdots\ \mathbf{v}_k^T]\mathbf{m}$ for $j\in[R+1]$.
    (By construction, such $(R+1)$ nodes exist and are distinct.)
From each of these $(R+1)$ nodes, ask for the respective bit
    $[\mathbf{v}_1^T\ \cdots\ \mathbf{v}_{z-1}^T\ \mathbf{u}_j^T\ \mathbf{v}_{z+1}^T\ \cdots\ \mathbf{v}_k^T]\mathbf{m}$ for $j\in[R+1]$.

***Response of any node to whom user connects:***
Upon receipt of a query for any data:
    If the node possesses that data, then return it.

***Decoding algorithm:***
For $j\in[R+1]$, download the bits
    $\mathbf{w}_j^T\mathbf{m}:=[\mathbf{v}_1^T\ \cdots\ \mathbf{v}_{z-1}^T\ \mathbf{u}_j^T\ \mathbf{v}_{z+1}^T\ \cdots\ \mathbf{v}_k^T]\mathbf{m}$ .
For $\ell\in[R]$, compute $\left(\mathbf{w}_\ell^T\mathbf{m}-\sum_{j=1}^{R+1}\mathbf{w}_j^T\mathbf{m}\right)$.
```

Theorem 2: The user can recover the desired record $z \in [k]$ of R bits by downloading the $(R+1)$ bits as described in Algorithm 2, and none of the nodes can obtain any information about z from the queries.

Proof: Recall that $\mathbf{u}_{R+1} = \sum_{i=1}^{R}\mathbf{u}_i$ and that R is assumed to be an even number. As a result,

$$\sum_{i=1}^{R+1}\mathbf{w}_i^T\mathbf{m} = [\mathbf{v}_1^T\ \cdots\ \mathbf{v}_{z-1}^T\ \ 0\ \ \mathbf{v}_{z+1}^T\ \cdots\ \mathbf{v}_k^T]\mathbf{m}\ .$$

It follows that for every $\ell \in [R]$,

$$\left(\mathbf{w}_\ell^T\mathbf{m} - \sum_{i=1}^{R+1}\mathbf{w}_i^T\mathbf{m}\right) = [0\ \cdots\ 0\ \mathbf{u}_\ell^T\ 0\ \cdots\ 0]\mathbf{m} = \mathbf{u}_\ell^T\mathbf{m}_z\ .$$

This gives the desired data.

To see privacy, consider any node who receives a query from the user, say a query for $[\mathbf{v}_1^T\ \cdots\ \mathbf{v}_{z-1}^T\ \mathbf{u}_j^T\ \mathbf{v}_{z+1}^T\ \cdots\ \mathbf{v}_k^T]\mathbf{m}$. One can see that in Algorithm 2, for any value of z, the probability of the node receiving this query is $\frac{1}{(R+1)^{k-1}}$. This probability is independent of z. The probability of the node not being queried is $\left(1 - \frac{1}{(R+1)^{k-2}}\right)$, which is also independent of z. Thus the node cannot obtain any information about the value of z. ■

When R is odd: Define $\mathbf{u}_{R+1} := \sum_{i=2}^{R}\mathbf{u}_i$. Apply the same encoding and PIR query construction. In the decoding algorithm, compute the bits $\left(\mathbf{w}_\ell^T\mathbf{m} - \sum_{j=2}^{R+1}\mathbf{w}_j^T\mathbf{m}\right)$ which will give the desired data.

Replicated storage: If all the records are stored in each of $n = (R+1)$ nodes, then Algorithm 2 can perform PIR by downloading just $(R+1)$ bits by sending the $(R+1)$ queries to these $(R+1)$ replicated nodes.

V. Code and PIR Algorithm with Linear Storage

As dictated by Theorem 1, the storage space used by the capacity-achieving construction of Section IV is super-linear in R and requires a connectivity (and hence the total query size) to be linear or higher in R. This section presents an explicit erasure code and PIR algorithm which has a linear storage overhead, a high reliability, and are a small factor away from capacity with respect to the download. The algorithm is associated with an additional parameter Δ: the user can perform PIR of any record by connecting to *any arbitrary* Δ of the n nodes. We assume that Δ is even and $\Delta \geq 2k$.

A. Data Encoding and Storage

The encoding is performed using the product-matrix framework of [15]. First, choose p to be any number such that $p \geq \lceil \log_2 n \rceil$. Each record is split into chunks of $\frac{\Delta-(k-1)}{2}p$ bits each.[1] Each chunk is operated upon independently and identically by the encoding and PIR algorithms. In what follows, we thus consider only a *single chunk of each record.* Each individual chunk of $\frac{\Delta-(k-1)}{2}p$ bits is represented as $\frac{\Delta-(k-1)}{2}$ symbols in the finite field $\mathbb{F}_{2^p}$.

Let Ψ be any $(n \times \frac{\Delta}{2})$ matrix[2] that satisfies (a) any $\frac{\Delta}{2}$ rows are linearly independent, and (b) when restricted to the first k columns, every k rows are linearly independent. For instance, one can choose Ψ to be a Vandermonde matrix. Denote the i^{th} row of Ψ by $\boldsymbol{\psi}_i^T$. Now, construct a $(\frac{\Delta}{2} \times \frac{\Delta}{2})$ *symmetric* matrix M, whose elements comprise all the symbols of all the records, arranged in a specific manner as in [15, Section IV].

For $i \in [n]$, the $\frac{\Delta}{2}$ symbols stored in node i are

$$\boldsymbol{\psi}_i^T M\ .$$

This completes the encoding procedure.

The code ensures that the first k nodes are *systematic*, i.e., the data stored in node $i \in [k]$ includes the i^{th} record in an uncoded form [15, Section IV-B].

B. PIR Algorithm

Algorithm 3 describes the PIR protocol. This protocol amalgamates the two-server PIR scheme of [5] with erasure codes.

[1]Assume that $(\Delta - (k-1))$ is even. The case of $(\Delta - (k-1))$ being odd can be handled easily via space sharing across two stripes.

[2]Recall that Δ is even.

Algorithm 3 PIR under Codes with Linear Storage Overhead

```
Query construction:
Let $z\in[k]$ denote the desired record.
Choose an $\frac{\Delta}{2}$-length vector $\mathbf{r}$ uniformly at random from $\mathbb{F}_{2^p}^{\Delta/2}$.
Connect to some $\Delta$ arbitrary nodes.
Pass the vector $\mathbf{r}$ to any $\frac{\Delta}{2}$ of these nodes.
Pass the vector $(\mathbf{r}+\boldsymbol{\psi}_z)$ to the $\frac{\Delta}{2}$ remaining nodes.

Response of any node $h$ to whom user connects:
Receive a vector $\mathbf{q}$ from the user.
Return the inner product of the stored data $\boldsymbol{\psi}_h^T M$ with $\mathbf{q}$, i.e.,
    return $\boldsymbol{\psi}_h^T M\mathbf{q}$.

Decoding algorithm:
Download $\{\boldsymbol{\psi}_{h_1}^T M\boldsymbol{r},\ldots,\boldsymbol{\psi}_{h_{\Delta/2}}^T M\boldsymbol{r}\}$ from the first set of
    $\frac{\Delta}{2}$ nodes connected to, where $h_1,\ldots,h_{\Delta/2}$ denote the
    identities of these nodes.
Using the property that any $\frac{\Delta}{2}$ rows of the $(n\times\frac{\Delta}{2})$ matrix $\Psi$
    are linearly independent, obtain $M\boldsymbol{r}$.
Download $\{\boldsymbol{\psi}_{h_{\Delta/2+1}}^T M(\mathbf{r}+\boldsymbol{\psi}_z),\ldots,\boldsymbol{\psi}_{h_\Delta}^T M(\mathbf{r}+\boldsymbol{\psi}_z)\}$ from
    the remaining $\frac{\Delta}{2}$ nodes connected to, where
    $h_{h_{\Delta/2+1}},\ldots,h_\Delta$ denote the identities of these nodes.
Using the property that any $\frac{\Delta}{2}$ rows of the $(n\times\frac{\Delta}{2})$ matrix $\Psi$
    are linearly independent, obtain $M(\mathbf{r}+\boldsymbol{\psi}_z)$.
Subtract the results of the previous steps to obtain $M\boldsymbol{\psi}_z$.
The matrix $M$ is symmetric, and hence this is the same as
    having $\boldsymbol{\psi}_z^T M$.
By construction, for any $z\in[k]$, the data $\boldsymbol{\psi}_z^T M$ contains the
    $z$th record.
```

Theorem 3: For any $z \in [n]$, Algorithm 3 allows the user can recover $\boldsymbol{\psi}_z^T M$, and none of the nodes can obtain any information about the value of z.

Proof: As shown in the algorithm, the user recovers the desired record at the end of the decoding procedure. We now prove privacy. Any node contacted by the user receives either the vector $\mathbf{r}$ or the vector $(\mathbf{r} + \psi_z)$. Since $\mathbf{r}$ is chosen independent of z, no information about z is revealed in the former case. In case of the latter, since $\mathbf{r}$ is also chosen uniformly at random from $\mathbb{F}_{2^p}^{\Delta/2}$, no information about z or ψ_z can be obtained from $(\mathbf{r}+\psi_z)$. Finally, since the choice of the Δ nodes is arbitrary, this choice also does not reveal any information. ■

C. Analysis

Storage overhead: The code partitions each record of size R bits into $\frac{2R}{(\Delta-(k-1))p}$ chunks of $\frac{\Delta-(k-1)}{2}p$ bits each. For each chunk, each node stores $\frac{\Delta}{2}p$ bits, and hence each node stores a total of $\alpha = R\frac{\Delta}{\Delta-(k-1)}$ bits. The only constraint on the total number of nodes n is that $n \geq 2\Delta$. Thus the number of nodes required is independent of R and is linear in k. For a fixed value of k, we can thus assume n to also be fixed. The code thus has a linear storage overhead:

$$\text{storage overhead} = \frac{\Delta}{\Delta-(k-1)}\frac{n}{k}. \quad (5)$$

In particular, when $n = \Delta = 2k$, storage overhead is ≤ 4.

Download during PIR: PIR entails a download of $p\Delta$ bits per chunk, and hence a total download of

$$\text{download} = 2R\frac{\Delta}{\Delta-(k-1)} \leq 4R \quad (6)$$

bits. This is a factor at most 4 away from capacity.

Reliability of the erasure code: Under this erasure code, the entire database can be recovered from *any* k of the n nodes [15, Theorem 3]. This gives the storage system an ability to tolerate the failure of any $(n-k)$ storage nodes without losing any data, thereby guaranteeing a high level of reliability.

VI. Conclusions and Open Problems

The topic of private information retrieval (PIR) has been fairly well explored in the literature on theoretical computer science and cryptography. However, to the best of our knowledge, all the previous works assume a replication-based setting. This paper considers PIR when data is stored using erasure codes (replication is a special case), establishing capacity with respect to the metric of download and constructing explicit codes and PIR algorithms. Optimizing other metrics remain open for PIR in the erasure-coded setting. For instance, the capacity-achieving example of Fig. 1 has maximum reliability (it is maximum-distance-separable) whereas the code of Section IV does not provide any reliability guarantees. It remains open as to what is the best possible reliability that a capacity-achieving erasure code can provide. Furthermore, it is still unclear whether erasure codes are better or worse for PIR than replication-based storage systems, and deriving an apples-to-apples comparison remains open.

References

[1] R. Henry, F. Olumofin, and I. Goldberg, "Practical PIR for electronic commerce," in *ACM conf. on Computer and comm. security*, 2011.

[2] G. Fanti, M. Finiasz, and K. Ramchandran, "One-way private media search on public databases: The role of signal processing," *IEEE Signal Processing Magazine*, 2013.

[3] "Stealth Software Inc. http://www.stealthsoftwareinc.com/."

[4] S. Yekhanin, "Private information retrieval," *Comm. of the ACM*, 2010.

[5] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *Journal of the ACM*, 1998.

[6] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *Proc. FOCS*, 1995.

[7] A. Ambainis, "Upper bound on the communication complexity of private information retrieval," *Automata, Languages and Programming*, 1997.

[8] Y. Gertner, S. Goldwasser, and T. Malkin, "A random server model for private information retrieval," *Rand. and Approx. Tech. in CS*, 1998.

[9] A. Beimel, Y. Ishai, E. Kushilevitz, and J. Raymond, "Breaking the $O(n^{1/(2k-1)})$ barrier for information-theoretic private information retrieval," in *FOCS*, 2002.

[10] D. Ford *et al.*, "Availability in globally distributed storage systems," in *USENIX OSDI*, 2010.

[11] "HDFS-RAID. http://wiki.apache.org/hadoop/HDFS-RAID."

[12] K. V. Rashmi *et al.*, "A solution to the network challenges of data recovery in erasure-coded distributed storage systems: A study on the Facebook warehouse cluster," in *Proc. USENIX HotStorage*, Jun. 2013.

[13] S. Hasan, Y. Ben-David, G. Fanti, E. Brewer, and S. Shenker, "Building dissent networks: Towards effective countermeasures against large-scale communications blackouts," in *FOCI*, 2013.

[14] N. B. Shah, K. V. Rashmi, and K. Ramchandran, "One extra bit of download ensures perfectly private information retrieval," 2014. [Online]. Available: arXiv

[15] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for the MSR and MBR points via a product-matrix construction," *IEEE Trans. Inf. Th.*, Aug. 2011.