Bounds on the Length of Functional PIR and Batch Codes

Yiwei Zhang
Dept. of Computer Science
Technion-Israel Inst. of Technology
Haifa 3200003, Israel
Email: ywzhang@cs.technion.ac.il

Eitan Yaakobi
Dept. of Computer Science
Technion-Israel Institute of Technology
Haifa 3200003, Israel
Email: yaakobi@cs.technion.ac.il

Tuvi Etzion
Dept. of Computer Science
Technion-Israel Inst. of Technology
Haifa 3200003, Israel
Email: etzion@cs.technion.ac.il

***Abstract*—A *functional k-PIR code* of dimension s consists of n servers storing linear combinations of s linearly independent information symbols. Any linear combination of the s information symbols can be recovered by k disjoint subsets of servers (the reason for this somehow abused definition will be explained in the sequel). The goal is to find the smallest number of servers for given k and s. We provide lower bounds on the number of servers and constructions which yield upper bounds. For $k \leq 4$ we provide exact bounds on the number of servers. Furthermore, we provide some asymptotic bounds. The problem coincides with the well known private information retrieval problem based on a coded database to reduce the storage overhead.**

If any multiset of size k of linear combinations from the linearly independent information symbols can be recovered by k disjoint subset of servers, then the servers form a *functional k-batch code*. A functional k-batch code is also a functional k-PIR, where all the k linear combinations in the multiset are equal. We provide some bounds on the number of servers for functional k-batch codes. In particular we present a random construction and a construction based on simplex codes, WOM codes, and RIO codes.

I. Introduction

A. General Background

A *Private Information Retrieval* (PIR) protocol allows a user to retrieve a data item from a database, in such a way that the servers storing the data will get no information about which data item was retrieved. The problem was introduced in [3]. For a set of k servers, the goal is to design an efficient k-server PIR protocol, where efficiency is measured by the total number of bits transmitted by all parties involved.

The classical model of PIR assumes that each server stores a copy of an s-bit database, so the *storage overhead*, namely the ratio between the total number of bits stored by all servers and the size of the database, is k. However, recent work combines PIR protocols with techniques from distributed storage (where each server stores only a coded fraction of the database) to reduce the storage overhead. This approach was first considered in [12], and several papers have developed this direction further. Our discussion on PIR will follow the breakthrough approach presented in [7], which shows that n servers (for some $n > k$) may emulate a k-server PIR protocol with storage overhead significantly lower than k. The scheme used for this purpose is called a *k-PIR* and will be discussed in the next paragraph.

The s-bit database $\mathcal{S}$ is considered as the information bits of a linear code of length n and dimension s. This code has an $s \times n$ generator matrix $\mathbf{G}$. The linear combinations related to the codeword $\mathcal{S} \cdot \mathbf{G}$ are stored in the n servers. In other words, the i-th server stores the linear combination generated when the s-bit information word is multiplied by the i-th column of $\mathbf{G}$. The generator matrix $\mathbf{G}$ represents a k-PIR scheme if there are k pairwise disjoint subsets of $[n] \triangleq \{1, 2, \dots, n\}$, $R_1, R_2, \dots, R_k$, such that the sum of the columns of $\mathbf{G}$ related to each such subset is the data item which the user wants to retrieve. Using these k subsets any known k-PIR protocol can be emulated with the given n servers. The advantage of this scheme is in reduced amount of storage used for a k-PIR protocol. The goal in the design of such a PIR scheme is to find the smallest n, given s and k. This problem was considered in several papers, e.g. [1], [7] and references therein.

In all the PIR protocols in the literature, the user wants to retrieve one out of the s information bits of the database. But, at least theoretically, it is quite natural that the user will want to retrieve a linear combination of the s bits of information symbols. This is not applicable in the PIR application, but it is applicable in related applications such as batch codes [9] or availability codes [10]. Hence, such a scheme will be called (with some abuse of the name) a *k-functional PIR code*.

A k-PIR code is a special case of a k-batch code. Batch schemes were introduced by Ishai et al. [9], motivated by different applications for load-balancing in storage and cryptographic protocols. Originally, batch codes were defined in a very general form, i.e., s information symbols are encoded into n-tuples of strings where each string is called a bucket. Each bucket contains a few linear combinations of the information symbols. A single user wants to retrieve a batch of k distinct data items (out of the s data items) by reading at most t symbols from each bucket. The goal in the design of a batch scheme is to find the smallest total length of all the buckets, given s, k, t and n.

A stronger variant of batch codes [9] is intended for a multi-user application instead of a single-user setting, known as the *multiset batch codes*. In this variant we have k different users each requesting a data item, where some of the requests are allowed to be the same. Therefore, all the k requests constitute a multiset of data items (each being one out of the s data items, replications allowed). Moreover, each bucket is allowed to be accessed by at most one user. A special case of multiset batch code is when each bucket contains only one symbol. This model is called a *primitive multiset batch code* [9] (or *k-batch code*). This family of batch codes is the most studied in the literature. In this setup a k-PIR code is a special case of a k-batch code. We restrict our definition only to this family of batch codes. They are represented by

978-1-5386-9291-2/19/$31.00 ©2019 IEEE

an $s \times n$ generator matrix $\mathbf{G}$. Such a matrix represents a k-batch scheme if there are k pairwise disjoint subsets of $[n]$, $R_1, R_2, \ldots, R_k$, such that the k sums from each subset of the columns in $\mathbf{G}$ constitute a multiset of data items which some k users want to retrieve. Hence, the requests in a k-PIR are a special case of the requests in a k-batch when the multiset contains only one specific item k times. Therefore, a k-batch code can always work as a k-PIR code but not vice versa. The goal in the design of a batch scheme is to find the smallest n, given s and k. This problem was considered in several papers, e.g. [1], [9], [11]. Similarly as our generalization of PIR into functional PIR, a batch code is generalized into a *functional batch code*.

A related family of codes to functional batch codes are *random I/O (RIO) codes*. This family of codes was recently introduced in [13] and provides a coding scheme to improve the random input/output performance of flash memories. An (n, M, t) RIO code stores t pages in n cells with $t+1$ levels such that it is enough to sense a single read threshold in order to read any of the t pages. In [13] it was shown that the design of RIO codes is equivalent to the design of *write-once memory* (*WOM*) *codes* [4], [8]. However, while in WOM codes, the messages are received one after the other and thus are not known all in advance, in RIO codes the information of all logical pages can be known in advance when programming the cells. This variant of RIO codes, called *parallel RIO codes*, was introduced in [14]. A recent construction of parallel RIO codes [15] used the coset coding scheme [4] with Hamming codes. This construction is equivalent to the requirements of functional batch codes. Hence, any functional batch code is also a parallel RIO code, but the other direction does not necessarily hold.

B. General Description of the Problem

Assume there are n servers, each storing a linear combination of s linearly independent items. Each of these s items will be called an *information symbol*. Each linear combination which consists of at least one of these information symbols will be called a *coded symbol*. There are k users who want to retrieve k linear combinations of items from these servers. Each such linear combination which a user wants to retrieve will be called a *request*. Each user has exactly one such request and he should approach a set of servers to obtain his request. The set of servers which are approached by two different users must be disjoint. We would like to know the smallest number of servers that is required to satisfy any k requests of the k users. This scheme will be called a *functional k-batch (code)*. If each request contains exactly one information symbol, then the scheme will be called a *k-batch code*.

If the k requests are the same (linear combination) then the scheme will be called a *functional k-PIR code* and furthermore if these k requests contain the same information symbol, then the scheme will be called a *k-PIR code*. This definition for k-PIR coincides with the definition for k-PIR given in [7] for a single user. Let $FB(s,k)$ ($B(s,k)$, $FP(s,k)$, $P(s,k)$, respectively) be the minimum number of servers required for s items and k requests for functional k-batch (k-batch, functional k-PIR, k-PIR, respectively).

A *functional k-batch code* of length n and dimension s consists of n servers and s information symbols $\{x_1, x_2, \ldots, x_s\}$. Each server stores a nontrivial linear combination of the information symbols (which are the coded symbols), i.e. the j-th server stores a linear combination Y_j, $1 \leq j \leq n$. For any request of k linear combinations $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k$ (not necessarily distinct) of the information symbols, there are k pairwise disjoint subsets $R_1, R_2, \ldots, R_k$ of $[n]$ such that the sum of the linear combinations in the related servers of R_j, $1 \leq j \leq k$, is $\boldsymbol{v}_j$, i.e. $\sum_{\ell \in R_j} Y_\ell = \boldsymbol{v}_j$. Each such $\boldsymbol{v}_i$ will be called a *requested symbol* and each such subset R_j will be called a *recovery set*. The functional k-batch code can be also represented by an $s \times n$ matrix $\mathbf{G}$ in which the j-th column has *ones* in positions $i_1, i_2, \ldots, i_\ell$ if and only if the j-th server stores the linear combination $x_{i_1} + x_{i_2} + \cdots + x_{i_\ell}$.

To summarize, a *k-batch code* is functional k-batch code, where each one of the requests $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$ is an information symbol. A *functional k-PIR code* is a functional k-batch code, where all the $\boldsymbol{v}_i$'s equal to one linear combination $\boldsymbol{v}$. A *k-PIR code* is functional k-PIR code, where the linear combination $\boldsymbol{v}$ contains exactly one information symbol.

C. Basic Results

Our goal is to obtain lower and upper bounds on $FB(s,k)$ and $FP(s,k)$. Some related bounds on $B(s,k)$ and $P(s,k)$ were derived or summarized in [1], [7], [11].

Lemma 1.

1) *For each* $s \geq 1$, $P(s, 2^{s-1}) = B(s, 2^{s-1}) = 2^s - 1$.
2) *When* k *is a fixed integer,* $P(s,k) = s + \Theta(\sqrt{s})$.
3) $B(s,k) = s + \Theta(\sqrt{s})$ *for* $k = 3, 4, 5$.
4) $B(s,k) = s + O(\sqrt{s} \log s)$ *for* $k \geq 6$.
5) $B(s, s^\varepsilon) \leq s + s^{7/8}$ *for* $7/32 \leq \varepsilon \leq 1/4$.
6) $B(s, s^\varepsilon) \leq s + s^{4\varepsilon}$ *for* $1/5 < \varepsilon \leq 7/32$.
7) $B(s,s) \leq 2s^{1.5}$.
8) $P(s, \sqrt{s}) = s + O(s^{(\log 3/2)})$.
9) $P(s, s^\varepsilon) = s + O(s^{0.5+\varepsilon})$, $0 < \epsilon < 1/2$.
10) $B(s, k = \Theta(s^\varepsilon)) = s + o(s)$, $0 < \epsilon < 1$.
11) $P(s, k = \Theta(s^\varepsilon)) = s + o(s)$, $0 < \epsilon < 1$.

We continue in the rest of the paper to derive lower and upper bounds on $FP(s,k)$ and $FB(s,k)$. For lack of space some constructions and proofs are omitted. They appear in the full version of this paper [16]. Simple bounds on $FB(s,k)$ and $FP(s,k)$ are given in the following two theorems.

Theorem 1. *If* s *and* k *are positive integers, then*

1) *For* $k > 1$, $FB(s,k) > FB(s,k-1)$.
2) *For* $k > 1$, $FP(s,k) > FP(s,k-1)$.
3) *For* $s \geq 1$, $FP(s,1) = FB(s,1) = s$.
4) *For* $s \geq 1$, $FP(s,2) = s+1$.
5) *For* $s \geq 1$ *and* $k \geq 1$, $FP(s,2k) = FP(s,2k-1)+1$.
6) *For* $s \geq 1$ *and* $k \geq 1$, $FP(s,k) \leq FB(s,k)$.

Theorem 2. *If* s, t, s_1, s_2, k_1, k_2 *are positive integers, then*

(1) $FP(s, 2^{s-1}) = 2^s - 1$.
(2) $FP(s, k_1 + k_2) \leq FP(s, k_1) + FP(s, k_2)$.
(3) $FP(s_1 + s_2, k) \leq FP(s_1, k) + FP(s_2, k)$.
(4) $FP(rt, 2^r) \leq 2t(2^r - 1)$.

II. A Construction of Functional PIR Codes

In this section an explicit construction of functional k-PIR codes, when k is a power of 2, is presented. The code which has tr information symbols will be represented by two $(t+1) \times 2^r$ arrays. One array will be defined in the construction and the second array will be defined in the proof for the correctness of the construction. In the first array, each entry, except for the entries of the last column, represents the content of different servers. The last column of the array contains *zeroes*. In the second array, each column represents a recovery set. The second array is obtained from the first array by a permutation defined via a translation induced from the requested symbol. By puncturing p times this code of length 2^r, a functional k-PIR codes for $k = 2^r - 2p$ will be obtained.

Construction 1. *Let $\{x_j^i : 1 \leq i \leq t,\ 1 \leq j \leq r\}$ be the set of $s = rt$ information symbols. Let $\mathcal{T}$ be a $(t+1) \times 2^r$ array whose last column consists of* zeroes*. The columns of $\mathcal{T}$ are indexed by the elements of the power set $2^{[r]}$. The i-th row, $1 \leq i \leq t$, contains the 2^r linear combinations of the symbols $\{x_j^i : 1 \leq j \leq r\}$. In particular, the entry on the column indexed by $A \in 2^{[r]}$ contains the linear combination $x_A^i = \sum_{j \in A} x_j^i$ (note that $x_\varnothing^i = 0$). Finally, the $(t+1)$-th row is a parity row, where the entry in the column indexed by A is $X_A = \sum_{i=1}^t x_A^i = \sum_{i=1}^t \sum_{j \in A} x_j^i$. This entry will be called the* leader *of the column. Note that only the entries of the column indexed by $\varnothing$ do not correspond to information stored in a server. The parity of this column which is* zero *is stored in the $(t+1)$-th row and it is also called a leader. Each other symbol in the array $\mathcal{T}$ is stored in a different server. The array $\mathcal{T}$ contains all the $n = (2^r - 1)(t+1)$ symbols and hence it will be called* the stored symbols array.

By Theorem 2, $FP(rt, 2^r) \leq 2t(2^r - 1)$. In the next theorem this upper bound is improved.

Theorem 3. *The code of Construction 1 is a functional 2^r-PIR code. Therefore, $FP(rt, 2^r) \leq (2^r - 1)(t+1)$.*

Proof: Let $\boldsymbol{v}$ be the requested symbol, i.e., $\boldsymbol{v}$ is a linear combination

$$\boldsymbol{v} = \boldsymbol{v}^1 + \boldsymbol{v}^2 + \cdots + \boldsymbol{v}^t,$$

where each $\boldsymbol{v}^i$ is a linear combination of the information symbols $\{x_j^i : 1 \leq j \leq r\}$, $1 \leq i \leq t$. We also define $\boldsymbol{v}^{t+1} = 0$.

Given the $(t+1) \times 2^r$ stored symbols array $\mathcal{T}$, we construct a new $(t+1) \times 2^r$ array $\mathcal{R}^{\boldsymbol{v}}$ as follows. The rows and the columns of $\mathcal{R}^{\boldsymbol{v}}$ are indexed exactly in the same way as the rows and columns of $\mathcal{T}$ are indexed. To the symbol in $\mathcal{T}$ in the entry on the i-th row, $1 \leq i \leq t+1$, and the column indexed by any subset A of $2^{[r]}$, we add $\boldsymbol{v}^i$ to obtain the corresponding symbol in $\mathcal{R}^{\boldsymbol{v}}$ in the same entry. The array $\mathcal{R}^{\boldsymbol{v}}$ will be called *the recovery array for* $\boldsymbol{v}$ since each column contains the content of the servers which form one of the recovery sets. Note, that the i-th row of $\mathcal{R}^{\boldsymbol{v}}$, $1 \leq i \leq t+1$, is a permutation of the i-th row of $\mathcal{T}$ and hence the symbols contained in $\mathcal{R}^{\boldsymbol{v}}$ are exactly the same symbols contained in $\mathcal{T}$, which implies that the information of each server is contained in exactly one entry of $\mathcal{R}^{\boldsymbol{v}}$, but usually not in the same entry as in $\mathcal{T}$. The exceptions are the $(t+1)$-th row and each row i for which $\boldsymbol{v}^i = 0$. It implies that the array $\mathcal{R}^{\boldsymbol{v}}$ represents the content of the servers, but in different entries from those of $\mathcal{T}$. We claim now that each column of $\mathcal{R}^{\boldsymbol{v}}$ contains a set of servers which form a recovery set.

Hence, to complete the proof it is sufficient to show that the sum of the symbols in each column of $\mathcal{R}^{\boldsymbol{v}}$ is $\boldsymbol{v}$. For a subset A of $[r]$ let $\mathcal{T}_A$ be the column of $\mathcal{T}$ indexed by A and let $\mathcal{R}_A^{\boldsymbol{v}}$ be the column of $\mathcal{R}^{\boldsymbol{v}}$ indexed by A. The sum of the symbols in $\mathcal{R}_A^{\boldsymbol{v}}$ is computed from the symbols of $\mathcal{T}_A$ and the request $\boldsymbol{v}$ as follows

$$\sum_{i=1}^t (x_A^i + \boldsymbol{v}^i) + X_A = \sum_{i=1}^t x_A^i + X_A + \sum_{i=1}^t \boldsymbol{v}^i = \sum_{i=1}^t \boldsymbol{v}^i = \boldsymbol{v}.$$

Therefore, each column of $\mathcal{R}^{\boldsymbol{v}}$ can serve as a recovery set for the requested symbol $\boldsymbol{v}$. Thus, the proof of the theorem is completed. ■

Construction 1 was further amended to obtain the following theorem

Theorem 4. *$FP(rt, 2^r - 2p) \leq (2^r - p - 1)t + 2^r - 2p - 1$, for $0 \leq p < 2^{r-2}$.*

III. Lower Bounds on the Length of PIR Codes

This section is devoted to lower bounds on the length of functional PIR codes. When the number of requests k is a fixed constant (more precisely $k = o(s)$), $P(s,k) = s + o(s)$ (see Lemma 1) and hence the research objective is to analyze the redundancy part $o(s)$. However, for functional PIR codes this is not the case. By using a counting argument it will be proved in this section that $FP(s,k)$ grows linearly in s for $k \geq 3$, i.e., $\lim_{s \to \infty} FP(s,k)/s \geq c$ for some constant c to be determined. Using another approach in this section, a better lower bound on $FP(s,3)$ and $FP(s,4)$ is derived. Codes for $k = 4$ in Construction 1 attain this bound and hence the bound is exact.

Theorem 5. *For a fixed even integer $k \geq 4$,*

$$\lim_{s \to \infty} \frac{FP(s,k)}{s} \geq \frac{1}{H(1/k)} ,$$

where $H(\cdot)$ is the binary entropy function defined by $H(p) = -p \log p - (1-p) \log (1-p)$.

Proof: Suppose there exists a functional k-PIR code of dimension s and length n. For each request $\boldsymbol{v}$, we have k disjoint recovery sets of $[n]$. The sum of the sizes of all these $k(2^s - 1)$ recovery sets is at most $n(2^s - 1)$. Hence, the average size of a recovery set should be at most $\frac{n}{k}$.

Consider all the subsets of $[n]$ of size at most $\lceil \frac{n}{k} \rceil + 1$. If each such subset is used as a recovery set for some request, then the average size of a recovery set is at least

$$\frac{\sum_{i=1}^{\lceil \frac{n}{k} \rceil + 1} i \binom{n}{i}}{\sum_{i=1}^{\lceil \frac{n}{k} \rceil + 1} \binom{n}{i}}. \tag{1}$$

By manipulation on the binomial coefficients we have

$$\binom{n}{\lceil \frac{n}{k} \rceil + 1} > \sum_{i=1}^{\lceil \frac{n}{k} \rceil - 1} (\lceil \frac{n}{k} \rceil - i) \binom{n}{i}. \tag{2}$$

By developing the numerator in (1) and plugging (2) in the process we obtain

$$\sum_{i=1}^{\lceil \frac{n}{k}\rceil+1} i\binom{n}{i} > \left\lceil\frac{n}{k}\right\rceil \sum_{i=1}^{\lceil \frac{n}{k}\rceil+1}\binom{n}{i}.$$

Now, we can evaluate the average in (1) as

$$\frac{\sum_{i=1}^{\lceil \frac{n}{k}\rceil+1} i\binom{n}{i}}{\sum_{i=1}^{\lceil \frac{n}{k}\rceil+1}\binom{n}{i}} > \frac{\lceil\frac{n}{k}\rceil \sum_{i=1}^{\lceil \frac{n}{k}\rceil+1}\binom{n}{i}}{\sum_{i=1}^{\lceil \frac{n}{k}\rceil+1}\binom{n}{i}} = \left\lceil\frac{n}{k}\right\rceil \geq \frac{n}{k},$$

which contradicts our proof that the average size of a recovery set is at most $\frac{n}{k}$.

Therefore, not all the subsets of $[n]$ of size at most $\lceil\frac{n}{k}\rceil+1$ are used as recovery sets, which implies that $\sum_{i=1}^{\lceil \frac{n}{k}\rceil+1}\binom{n}{i} > k(2^s-1)$. The left hand side tends to $2^{nH(1/k)}$ as n tends to infinity. Hence, if $n = c\cdot s$, then

$$2^{csH(1/k)} > k(2^s-1),$$

which implies that $cH(1/k) > 1$ and the claim of the theorem follows. ■

The technique used in the proof of Theorem 5 can be applied slightly differently to obtain lower bounds on $FP(s,k)$ for specific parameters s and k.

Suppose we have a functional k-PIR code with dimension s and length n. For each request $\boldsymbol{v}$, we have k disjoint subsets of $[n]$, $R_1,\ldots,R_k$, where each one of them is a recovery set for $\boldsymbol{v}$. For each such request $\boldsymbol{v}$ we choose arbitrarily such k recovery sets. Therefore, $k(2^s-1)$ distinct recovery sets are chosen. Let $\Lambda(s)$ be the sum of the size of all these recovery sets. The k recovery sets $R_1,\ldots,R_k$ for any request $\boldsymbol{v}$ are pairwise disjoint, and hence

$$\sum_{i=1}^{k}|R_i| \leq n\ ,$$

which implies that

$$\Lambda(s) \leq n(2^s-1)\ . \tag{3}$$

On the other hand, a lower bound on $\Lambda(s)$ can be obtained by choosing the recovery sets with smallest size as possible. Let d be the largest integer such that

$$\sum_{i=1}^{d}\binom{n}{i} \leq k(2^s-1)\ . \tag{4}$$

It implies that in the chosen $k(2^s-1)$ recovery sets all the subsets of size at most d are included for the lower bound, and also $k(2^s-1)-\sum_{i=1}^{d}\binom{n}{i}$ subsets of size $d+1$. Thus,

$$\sum_{i=1}^{d} i\binom{n}{i} + (d+1)\left(k(2^s-1)-\sum_{i=1}^{d}\binom{n}{i}\right) \leq \Lambda(s). \tag{5}$$

The lower bound on $FP(s,k)$ is obtained by comparing (3) and (5), i.e., finding the minimum n for which

$$\sum_{i=1}^{d} i\binom{n}{i} + (d+1)\left(k(2^s-1)-\sum_{i=1}^{d}\binom{n}{i}\right) \leq n(2^s-1).$$

Example 1. *When k is even we have $FP(2,k) \leq \frac{3k}{2}$ (encode the two information symbols x_1 and x_2 into x_1, x_2, and x_1+x_2; each one of these three encoded symbol will appear $\frac{k}{2}$ times in the code.)*

Assume now that $n = FP(2,k) \leq \frac{3k}{2}-1$ and apply (5) for $s=2$, k and $n=\frac{3k}{2}-1$. For each request, three recovery sets are required for a total of $3k$ recovery sets. There are at most $n=\frac{3k}{2}-1$ recovery sets of size 1. Therefore, there are at least $\frac{3k}{2}+1$ recovery sets whose size is at least two. Hence, by (5),

$$(\frac{3k}{2}-1)+2\cdot(\frac{3k}{2}+1) = 3\cdot\frac{3k}{2}+1 \leq \Lambda(2)\ .$$

By (3), $\Lambda(2) \leq n\cdot(2^2-1) = 3\cdot\frac{3k}{2}-3$, a contradiction.

Therefore, $FP(2,k) > \frac{3k}{2}-1$ and thus $FP(2,k)=\frac{3k}{2}$ when k is even.

In a different approach of analyzing the number of ways to choose the recovery sets leads to the following theorem.

Theorem 6. *For any given $s \geq 3$ and $i = 0,1$, we have that*

$$FP(s,3+i) \geq \begin{cases} \frac{3}{2}s+2+i & \textit{if } s \textit{ is even} \\ \frac{3}{2}(s+1)+i & \textit{if } s \textit{ is odd} \end{cases}.$$

By Theorem 6, Theorem 3, and Theorem 1, we have

Corollary 1. *For any $t \geq 2$, $FP(2t,3) = 3t+2$, $FP(2t,4) = 3t+3$, $3t+3 \leq FP(2t+1,3) \leq 3t+4$ and $3t+4 \leq FP(2t+1,4) \leq 3t+5$.*

IV. A Random Construction for Batch Codes

In this section a random construction of functional batch codes is presented. This relies on a random construction for linear covering codes.

Definition 1. *For a binary code $\mathcal{C}$ of length n, the* covering radius *is the smallest integer R such that for any $\boldsymbol{v} \in \mathbb{F}_2^n$, there exists $\boldsymbol{u} \in \mathcal{C}$ such that $d(\boldsymbol{v},\boldsymbol{u}) \leq R$.*

Proposition 1. *[5] If $\mathcal{C}$ is a binary linear code of length n, and dimension k, with a parity check matrix $\mathbf{H}$, then $\mathcal{C}$ has covering radius R if and only if every column vector $\mathbb{F}_2^{n-k}$ is the sum of at most R columns of $\mathbf{H}$.*

Let $V(n,R)$ be the size of the Hamming ball of radius R. A code with covering radius R has at least $\frac{2^n}{V(n,R)}$ codewords and thus a linear code with covering radius R has dimension $k \geq n - \log V(n,R)$. Blinovskii [2] proved that almost all linear codes attain this sphere covering bound.

Theorem 7. *Let $0 \leq \rho < 1/2$, $\mathcal{C}_{k,n}$ be the ensemble of 2^{kn} linear codes generated by all possible binary $k\times n$ matrices, and $R_n = \lfloor \rho n \rfloor$. There exists a sequence k_n for which*

$$k_n/n \leq 1 - H(\rho) + O(n^{-1}\log n),$$

such that the fraction of codes $C_n \in \mathcal{C}_{k_n,n}$ which have covering radius R_n tends to 1, when n tends to infinity.

In other words, Theorem 7 implies that if a random binary matrix $\mathbf{H}$ of size $s\times n$ is considered as a parity check matrix of a linear code, then the covering radius $R=\rho n$ of the code satisfies $H(\rho) \sim \frac{s}{n}$ with probability tending to 1, when n tends to infinity, i.e., any column vector of length s is the sum of at most R columns of $\mathbf{H}$.

This is combined with a result of Cooper [6] on the invertibility of random binary matrices, to a random construction of functional batch codes.

Theorem 8. *If $c_1 = \frac{1}{2}$ and c_{k+1} is the root of the equation $H(z) = H(c_k) - zH(c_k)$, then*

$$\lim_{s\to\infty} \frac{FB(s,k)}{s} \leq \frac{1}{H(c_k)}.$$

A related lower bound on $FB(s,k)$ is derived as follows.

Theorem 9.

$$\lim_{s\to\infty} \frac{FB(s,k)}{s} \geq \frac{k}{\log(k+1)}.$$

Proof: Assume there is a functional k-batch code of length n and dimension s, represented by an $s \times n$ matrix $\mathbf{G}$. For any recovery process of a request $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k)$ with k vectors of length s, assign a label to each column of $\mathbf{G}$. The label is either 0 or some i, $1 \leq i \leq k$. A label 0 indicates that the column is not used in the recovery process of $\boldsymbol{v}$. A label i, indicates that the column is used in the recovery set for $\boldsymbol{v}_i$. Then the labeling of $\mathbf{G}$ for the request $\boldsymbol{v}$ is an element in $\{0,1,\ldots,k\}^n$. For any two different ordered k-tuples of request vectors $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k)$ and $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k)$, where $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$ are distinct k vectors and $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$ are also distinct k vectors, the labeling of $\mathbf{G}$ must be different. Therefore, $(k+1)^n \geq \binom{2^s-1}{k} k!$.

Thus,

$$\lim_{s\to\infty} \frac{n}{s} \geq \frac{k}{\log(k+1)}.$$

■

V. Simplex Codes as Functional Batch codes

Definition 2. *A $[2^r-1, r]$ simplex code is a linear code whose $r \times n$ generator matrix $\mathbf{G}$ contains each nonzero column vector of length r as a column.*

Simplex codes were used as write-once memory (WOM) codes and random I/O (RIO) codes. An $[n,k,t]$ WOM code is a coding scheme comprising of n binary cells such that it is possible to write a k-bit message t times while on each write the cell values can only change from zero to one.

In *linear WOM codes* [4] a binary matrix is used to encode messages by the syndromes of parity check matrices of error-correcting codes. Simplex codes were used by Godlewski [8], to show the existence of $[2^r-1, r, 2^{r-2} + 2^{r-4}+1]$ WOM codes.

Similarly to the conjecture on RIO codes raised and verified for $r = 3, 4$ in [15] we have the following conjecture which we verified for $r = 5$. Recall, that a functional batch code is a RIO code, but a RIO code might not be a functional batch code.

Conjecture 1. *The $[2^r-1, r]$ simplex code is a functional 2^{r-1}-batch code and therefore $FB(r, 2^{r-1}) = 2^r - 1$.*

The main idea of the construction of $[2^r-1, r, 2^{r-2} + 2^{r-4}+1]$ WOM codes by Godlewski [8] with simplex codes works as follows.

1) The first request $\boldsymbol{v}$ is simply satisfied by using $\boldsymbol{v}$ itself.
2) As long as there are at least 2^{r-1} nonzero available vectors, each request $\boldsymbol{v}$ can always be satisfied by finding a pair $\{\boldsymbol{u}, \boldsymbol{u}+\boldsymbol{v}\}$. This process can satisfy at least 2^{r-2} more requests and only stops when the number of unused vectors is less than 2^{r-1}.
3) The key part of Godlewski's construction is that it is still possible to find recovery sets of size four unless the number of unused vectors is less than 2^{r-2}. Thus, 2^{r-4} additional write requests can be satisfied.

To summarize, simplex codes can be used to satisfy roughly any $\frac{5}{16}2^r$ write requests, when considered as WOM codes. Since in the functional batch setting (or in parallel RIO codes) we know all the requests in advance, it is possible to make use of this knowledge and improve upon the $2^{r-2} + 2^{r-4} + 1$ result. This improvement comes either from the choice of many recovery sets of size one, or from a predetermined usage of the 2^{r-2} remaining vectors in Godlewski's method. Namely, we prove:

Theorem 10. *The $[2^r-1, r]$ simplex code is a functional $(2^{r-2} + 2^{r-4} + \lfloor \frac{2^{r/2}}{\sqrt{24}} \rfloor)$-batch code.*

Acknowledgment

E. Yaakobi and Y. Zhang were supported by the ISF grant 1817/18, T. Etzion and Y. Zhang by the BSF-NSF grant 2016692, and Y. Zhang was also by a Technion Fellowship. This research was partially supported by the Technion Hiroshi Fujiwara cyber security research center and the Israel cyber directorate.

References

[1] H. Asi and E. Yaakobi, *Nearly optimal constructions of PIR and batch codes*, *IEEE Trans. Inform. Theory*, vol. 65, no. 2, pp. 947–964, Feb. 2019.

[2] V. M. Blinovskii, *Asymptotically exact uniform bounds for spectra of cosets of linear codes*, *Problemy Peredachi Informatsii*, vol. 26, No. 1, pp. 99–103, 1990. Translated in: Problems of Inform. Transm., vol. 26, no. 1, pp. 83–86.

[3] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, *Private information retrieval*, *J. ACM*, vol. 45, pp. 965–981, 1998.

[4] G.D. Cohen, P. Godlewski, and F. Merkx, *Linear binary code for write-once memories*, *IEEE Trans. Inform. Theory*, vol. 32, no. 5, pp. 697–700, Oct. 1986.

[5] G. Cohen, M. Karpovsky, H. Mattson, Jr. and J. Schatz, *Covering radius: Survey and recent results*, *IEEE Trans. on Inform. Theory*, vol. 31, no. 3, pp. 328–343, May 1985.

[6] C. Cooper, *On the rank of random matrices*, *Random Structures Algorithms*, vol. 16, pp. 209–232, 2000.

[7] A. Fazeli, A. Vardy, and E. Yaakobi, *Private information retrieval without storage overhead: coding instead of replication*, *arxiv.org/abs/1505.0624*, May 2015.

[8] P. Godlewski, *WOM-codes construits à partir des codes de Hamming*, *Discrete Math.*, vol. 65, no. 3, pp. 237–243, Jul. 1987.

[9] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, *Batch codes and their applications*, *in Proc. of the 36-sixth Annual ACM Symposium on Theory of Computing*, pp. 262-271, Chicago, ACM Press, 2004.

[10] A. S. Rawat, D. S. Papailiopoulos, A. G. Dimakis, and S. Vishwanath, *Locality and availability in distributed storage*, *IEEE Trans. Inform. Theory*, vol. 62, no. 8, pp. 4481–4493, Aug. 2016.

[11] A. S. Rawat, Z. Song, A. G. Dimakis, and A. Gál, *Batch codes through dense graphs without short cycles*, *IEEE Trans. Inform. Theory*, vol. 62, no.4, pp. 1592–1604, Apr. 2016.

[12] N. Shah, K. Rashmi, and K. Ramchandran, *One extra bit of download ensures perfectly private information retrieval*, *IEEE Int. Symp. Inf. Theory (ISIT)*, pp. 856–860, 2014.

[13] E. Sharon and I. Alrod, *Coding scheme for optimizing random I/O performance*, *Non-Volatile Memories Workshop*, San Diego, Apr. 2013.

[14] E. Yaakobi and R. Motwani, *Construction of random input-output codes with moderate block lengths*, *IEEE Trans. on Comm.*, vol. 64, no. 5, pp. 1819–1828, May 2016.

[15] A. Yamawaki, H. Kamabe, and S. Lu, *Construction of parallel RIO codes using coset coding with Hamming code*, *IEEE Inf. Theory Workshop (ITW)*, pp. 239–243, Kaohsiung, Taiwan, Nov. 2017.

[16] Y. Zhang, E. Yaakobi, and T. Etzion, *Bounds on the length of functional PIR and batch codes*, *arxiv.org/abs/1901.01605*, Jan. 2019.