Convertible Codes: Enabling Efficient Conversion of Coded Data in Distributed Storage

Francisco Maturana, *Student Member, IEEE*, and K. V. Rashmi, *Member, IEEE*

***Abstract*—Erasure codes are essential for providing efficient resilience against node failures in distributed storage. Typically, an $[n,k]$ erasure code encodes k symbols into n symbols which are then stored in different nodes. Recent work by Kadekodi et al. shows that the failure rates of storage nodes vary significantly over time, and that changing the rate of the code (via a change in n and k) in response to such variations provides substantial storage space savings. However, the resource overhead of re-encoding the already encoded data is prohibitively high. We present a new theoretical framework formalizing *code conversion*—the process of converting data encoded with an $[n^I, k^I]$ code into data encoded with an $[n^F, k^F]$ code while maintaining desired decodability properties. We then introduce *convertible codes*, a new class of codes that allow for code conversions in a resource-efficient manner. This paper begins the study on convertible codes by focusing on linear MDS codes and the access cost of conversion. We derive a lower bound on the access cost of conversion and present an explicit optimal construction matching this bound for an important subclass of conversions. Additionally, we propose constructions with low field-size requirement for a broad subset of parameters. Our results show that it is possible to achieve code conversions with significantly less resources than the default approach of re-encoding for a wide range of parameters.**

***Index Terms*—Convertible codes, storage codes, coding theory, distributed storage systems, re-encoding.**

I. Introduction

Erasure codes have become an essential tool for protecting against node failures in distributed storage systems [2]–[8]. Under erasure coding, a set of k data symbols to be stored is encoded using an $[n,k]$ code to generate n coded symbols, called a *codeword* (or *stripe*). Each of the n symbols in a codeword is stored on a different storage node, and the system as a whole typically contains several independent codewords distributed across different subsets of storage nodes in the cluster.

Manuscript received July 31, 2021; revised January 6, 2022; accepted January 21, 2022. Date of publication March 2, 2022; date of current version June 15, 2022. This work was supported in part by NSF CAREER Award under Grant 19434090, in part by NSF CNS under Grant 1956271, in part by the Google Faculty Research Award, and in part by the Facebook Distributed Systems Research Award. An earlier version of this paper was presented in part at the 11th Innovations in Theoretical Computer Science Conference (ITCS) [1] [DOI: 10.4230/LIPIcs.ITCS.2020.66]. *(Corresponding author: Francisco Maturana.)*

The authors are with the Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: fmaturan@cs.cmu.edu; rvinayak@cs.cmu.edu).

Communicated by A. Thangaraj, Associate Editor for Coding and Decoding.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIT.2022.3155972.

Digital Object Identifier 10.1109/TIT.2022.3155972

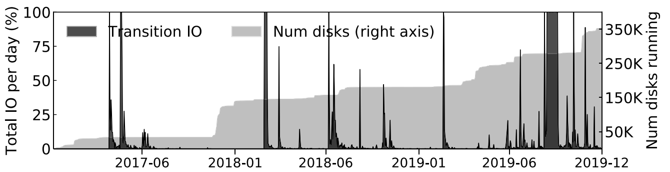


Fig. 1. (From [10]) The left y-axis shows the percentage of disk IO utilized by conversion (called "transition" in [10]) against time simulated from a trace of a production Google cluster. The right y-axis shows the size of the cluster in number of disks against time. Code conversions can result in big spikes in disk IO consumption that can overwhelm the cluster for several days.

A key factor that determines the choice of parameters n and k is the failure rate of the storage devices. It has been shown that failure rates of storage devices in large-scale storage systems can vary significantly over time and that changing the code rate, by changing n and k, in response to these variations yields substantial savings in storage space and hence the operating costs [9]. For example, in [9], the authors show that 11% to 44% reduction in storage space can be achieved by tailoring n and k to changes in observed device failure rates. Such a reduction in storage space requirement translates to significant savings in the cost of resources and energy consumed in large-scale storage systems. It is natural to think of potentially achieving such a change in code rate by changing only n while keeping k fixed. However, due to several practical system constraints, changing code rate in storage systems often necessitates change in both the parameters n and k [9]. We refer the reader to [9] for a more detailed discussion on the practical benefits and constraints of adapting the erasure-code parameters to the variations in failure rates in storage systems.

Changing n and k for codewords in a storage system, from $[n^I, k^I]$, to $[n^F, k^F]$, would involve converting already encoded data from one code to another. Clearly, it is always possible to *re-encode* the data in a codeword according to a new code by accessing (and decoding if necessary) all the original message symbols. However, such an approach, which we call the *default approach*, requires accessing a large number of symbols (for example, for MDS codes, the initial value of k number of symbols need to be accessed from each codeword), reading out all the data, transferring over the network, and re-encoding. Such conversions can generate a large amount of load on cluster resources, which adversely affects the foreground operations of the cluster. Figure 1 shows the IO load that would be caused by code conversions on a Google cluster with multiple hundreds of thousands of disks [10]. As seen from the figure, IO load from conversions can easily overwhelm the cluster for long periods of time. Furthermore, in some cases conversions might need to be performed in an expedited manner, for example, to avoid the risk of data loss when facing an unexpected rise in failure rate.

High IO load is problematic for such conversions because it slows down conversion as well as other important cluster processes, such as serving client requests. While recent work [10] has initiated a study on systems techniques to mitigate the spikes in the IO load caused by conversions, the total amount of work necessary for conversion still remains considerably high and these systems techniques introduce restrictions on other operations of the cluster such as data placement.

Given that the root cause of the problem is the high resource overhead involved in performing conversions on the underlying code, in this paper, we investigate the problem from a fundamental theoretical perspective.

There are also several other reasons to perform code conversions in storage systems. One may convert data that is frequently read into a code with a small k (in order to improve the performance of reconstructions) and convert data that is infrequently read into a code with large k (to achieve lower storage overhead). In addition, code conversions may need to be performed to keep the total size of the encoded data under a given threshold, or to maximize the reliability given the available storage space.

To the best of our knowledge, the existing literature [11]–[14] which formally studies the problem of changing the length and dimension of already encoded data does so from the perspective of the so-called *scaling problem*. The scaling problem [11] refers to the problem of evenly redistributing each codeword in a distributed storage system when additional nodes are added to the system and the level of failure tolerance (specifically, $(n-k)$) is kept constant. Some works [12], [14] generalize the scaling problem to broader cases where $(n-k)$ need not remain constant. However, even in cases where the scaling problem could be used to perform code conversion, it has several drawbacks that make it inefficient for conversion. For example, using the approach of scaling to achieve conversion requires accessing every symbol in each codeword and performing a significant amount of data movement to keep the amount stored in each node the same. While these costs are necessary to fulfill the goals of the scaling problem, they are unnecessary to achieve code conversion, making this approach inefficient. A more detailed discussion of the scaling problem and other related work is provided in Section II-A.

In this paper, we propose a theoretical framework to model the code conversion problem. Our approach is based on the insight that the problem of changing code parameters in a storage system can be viewed as converting *multiple codewords* of an $[n^I, k^I]$ code (denoted by $\mathcal{C}^I$) into (potentially multiple) codewords of an $[n^F, k^F]$ code (denoted by $\mathcal{C}^F$),[1] with desired constraints on decodability, such as both codes satisfying the maximum distance separability (MDS) property.

[1]The superscripts I and F stand for initial and final respectively, representing the initial and final state of the conversion.

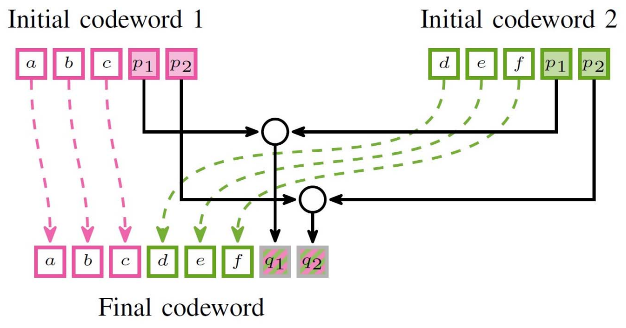


Fig. 2. Example of code conversion: two codewords of a $[5,3]$ MDS code are converted into one codeword of a $[8,6]$ MDS code. Unshaded boxes represent data symbols, and shaded boxes represent parity symbols. Some of the initial symbols are kept unchanged in the final codewords, as shown by the dashed arrows. Some initial symbols are read and downloaded (solid arrows). The downloaded data is then used to compute and write the remaining symbols in the final codewords.

To address the problem of code conversion, we then introduce a new class of codes, which we call *convertible codes*, that allow for resource-efficient conversions. The general formulation of code conversions provides a powerful framework to theoretically study convertible codes.

We now present an example to elucidate the concept of code conversion in the convertible codes framework.

Example 1: Consider conversion from an $[n^I = 5, k^I = 3]$ code $\mathcal{C}^I$ to an $[n^F = 8, k^F = 6]$ code $\mathcal{C}^F$. We will focus on the number of symbols read, i.e. *read access cost*, and on the number symbols written, i.e. *write access cost*, for conversion. The default approach to conversion is to read $k^I = 3$ symbols from each of the two initial codewords belonging to $\mathcal{C}^I$, decoding the original data, and using it to write two symbols of the final codeword belonging to $\mathcal{C}^F$, while keeping the three read symbols from each initial codeword unchanged as symbols of the final codeword. Thus, the default approach has a read access cost of 6 and write access cost of 2.

In the convertible codes framework, this conversion is achieved by converting two codewords of the initial code into a single codeword of the final code, as depicted in Figure 2. This approach uses specially designed systematic codes $\mathcal{C}^I$ and $\mathcal{C}^F$. Let $\mathbb{F}_q$ be the finite field of size $q = 37$. Let $a, b, c \in \mathbb{F}_q$ be the data symbols of the first initial codeword, $d, e, f \in \mathbb{F}_q$ be the data symbols of the second initial codeword. Let $p_1, p_2 : \mathbb{F}_q^3 \to \mathbb{F}_q$ be the parity functions for the initial code $\mathcal{C}^I$, and $q_1, q_2 : \mathbb{F}_q^6 \to \mathbb{F}_q$ be the parity functions for the final code $\mathcal{C}^F$. The parity functions are chosen as below:

$$p_1(a,b,c) = a + b + c, \quad p_2(a,b,c) = a + 2b + 4c.$$

This is an example of the general construction presented in Section V. The conversion procedure keeps the data symbols from each initial codeword unchanged in the final codeword, and then constructs the first (resp. second) parity of the final codeword as a linear combination of the first (resp. second) parity of each initial codeword. The final parity functions are chosen to satisfy the equation below:

$$q_1(a,b,c,d,e,f) = p_1(a,b,c) + p_1(d,e,f),$$
$$q_2(a,b,c,d,e,f) = p_2(a,b,c) + 8p_2(d,e,f).$$

It is straightforward to check that the initial and final codes defined by these parity functions have the MDS property. This conversion procedure requires reading two symbols from each initial codeword and writing two symbols, resulting in a total read access cost of 4 and a write access cost of 2, a reduction of $33.\overline{3}\%$ in the read access cost as compared to the default approach. This is also the minimum possible read cost, as will be shown in Section IV. ▶

In this paper, we begin the exploration of convertible codes by focusing on convertible codes which are *linear and MDS*. The properties of linear MDS codes have been well studied, and they are widely used in practice due to their relative simplicity and their storage overhead. As such they constitute a good starting point for studying convertible codes. Furthermore, we focus on the *access cost* of conversion. Access cost corresponds to the total number of symbols that are either read or written during conversion, which is a fundamental quantity that directly affects important metrics such as network bandwidth, IO and CPU resource usage.

Within the class of linear MDS codes, we focus on an important subclass of conversions, which we refer to as the *merge regime*. The merge regime corresponds to conversions where multiple codewords are merged into a single codeword. In other words, $k^F = \varsigma k^I$ for some integer $\varsigma \geq 2$, with arbitrary values of n^I and n^F. Using the convertible codes framework, we prove a tight lower bound on the access cost of conversions for linear MDS codes in the merge regime (Section IV). This lower bound identifies a broad region where significant savings in access cost are possible, and shows that in its complement region, linear MDS codes cannot achieve lower access cost than the default approach.

We cast the problem of constructing optimal convertible codes into a problem of constructing matrices which satisfy some special structural properties. Using this insight, we first propose a simple construction (Section V) which works for all parameters in the merge regime, but requires a large field size (exponential in the lengths of the codes). To address this issue, we introduce a sequence of constructions of convertible codes that have significantly lower field size requirements (Section VI). Our main results are summarized in Table I.

Throughout our analysis of convertible codes, we assume the values of the parameters (n^I, k^I) and (n^F, k^F) are known and fixed. However, in practice the value of (n^F, k^F) might not be known at the time of code construction, since it depends on the future failure rates of storage devices. To address this problem, we also show that the constructions presented in this paper can support access-optimal conversion for multiple possible values of (n^F, k^F) simultaneously.

TABLE I
MAIN RESULTS FOR THE ACCESS COST OF CONVERSIONS IN THE MERGE REGIME, I.E. A $(n^I, k^I; n^F, k^F = \varsigma k^I)$ CONVERTIBLE CODE

Approach	Access cost
Default	$\varsigma k^I + r^F$
Optimal ($r^I < r^F$)	$\varsigma k^I + r^F$
Optimal ($r^I \geq r^F$)	$\varsigma \min\{k^I, r^F\} + r^F$

Construction	Supported parameters	Field size requirement
Default approach	All parameters	$\max\{n^F - 1, n^I - 1\}$
General (Section V)	All parameters	$\max\left\{\begin{smallmatrix} 2^{\mathcal{O}((n^F)^3)}, \\ n^I - 1 \end{smallmatrix}\right\}$
Hankel-I (Section VI)	$r^F \leq \lfloor r^I / \varsigma \rfloor$	$\max\{n^F - 1, n^I - 1\}$
Hankel-II (Section VI)	$r^F \leq r^I - \varsigma + 1$	$\max\{k^I r^I, n^I - 1\}$
Hankel$_s$ for $\varsigma \leq s \leq r^I$ (Section VI)	$r^F \leq (s - \varsigma + 1)\lfloor r^I / s \rfloor + \max\{(r^I \bmod s) - \varsigma + 1, 0\}$	$\max\left\{\begin{smallmatrix} s k^I + \lfloor r^I / s \rfloor - 1, \\ n^I - 1 \end{smallmatrix}\right\}$

II. RELATED WORK, BACKGROUND AND NOTATION

In this section, we place convertible codes within the larger context of traditional codes and more recent works on codes for distributed storage. Then, we review some basic concepts and notation that will be used throughout the paper.

A. Related Work

MDS erasure codes, such as Reed-Solomon codes [15], are widely used in storage systems because they achieve the optimal tradeoff between failure tolerance and storage overhead [16], [17]. However, the use of erasure codes in storage systems raises a host of other aspects to optimize for. Several works in the literature have studied these aspects and proposed codes that optimize them.

One aspect of storage codes that received considerable attention early on is the computational overhead involved in encoding and decoding of data. *Array codes* [18]–[21] are designed to use XOR operations exclusively, which are typically faster to execute, and aim to decrease the complexity of encoding and decoding.

Another aspect of storage codes that has received considerable attention in the recent past is related to the resource overhead associated with repair of failed nodes. Several approaches have been proposed to alleviate this problem. Dimakis *et al.* [22] proposed a new class of codes called *regenerating codes* that minimize the amount of network bandwidth consumed during repair operations. Under the regenerating codes model [22], each symbol (i.e., node) is represented as an α-dimensional vector over a finite field. During repair of a failed node, download of elements of the finite field (i.e., "sub-symbols") is allowed as opposed to the whole vector (i.e., one "entire" symbol). This line of research has led to several constructions [23]–[42], generalizations [43]–[45], and more efficient repair algorithms for Reed-Solomon codes [46]–[53]. Several of these constructions [26], [35], [54]–[56] minimize the amount of IO consumed during repairs in addition to minimizing the network bandwidth consumption. It has been shown that meeting the lower bound on the network bandwidth required by repair when MDS property and high rate are desired necessitates large sub-packetization [54], [56]–[58], which negatively affects certain key performance metrics in storage systems [6], [7]. To overcome this issue, several works [59], [60] have proposed code constructions that relax the requirement of meeting lower bounds on IO and bandwidth for repair operations in order to reduce the degree of sub-packetization.

The challenge of code repair has also been addressed by another class of codes, called *locally repairable codes* (LRCs) [61]–[77]. These codes focus on the locality of codeword symbols during repair, that is, the number of nodes that need to be accessed when repairing a single failure. LRCs improve repair performance, since missing information can be recovered by accessing a small subset of symbols. LRCs and convertible codes optimized for access cost both aim to minimize the number of symbols that need to be accessed, albeit for different operations in storage systems.

Recent literature on storage codes has also considered the problem of redistributing data when additional devices are added to a distributed storage system, which is known as the *scaling problem* [11], [12], [14], [78]–[86]. The setting considered consists of an n node distributed storage system where the data is encoded using an $[n, k]$ MDS code, where the n symbols of each codeword are spread across evenly on all the n nodes in the system. Then, s new empty nodes are added to the system, and the data (which was encoded under an $[n, k]$ MDS code) needs to be updated to an $[n' = n + s, k' = k + s]$ MDS code. The central goal of this problem is to evenly redistribute each codeword across all n' nodes while reducing the total amount of data transferred across nodes and ensuring the MDS property holds. In some cases, it is additionally required that the ratio of data to parity in each node is the same (e.g. [83]). Some works consider more general scaling scenarios: for example [14] considers the case where $k < k'$ and $n < n'$, and [12] considers arbitrary $n' > k'$. The scaling problem is fundamentally different from the conversion problem that we study in this paper because of the need to evenly redistribute data across nodes under scaling. Hence, some of the key constraints and limitations of the scaling problem do not apply to code conversion. For example, scaling necessitates modifying every node in the system (incurring a high access cost) and necessitates transfer of data not for the purpose of conversion (i.e. changing n and k) but for the purpose of rebalancing the amount of data stored by each codeword in a given node. On the other hand, under the code conversion problem, we do not impose any requirements on data balancing. This is because, typically, large-scale distributed storage systems balance data across nodes at a higher level rather than at the level of each codeword [2], [5].

Several works have studied scenarios where encoded data is transformed to conform to a different code. In [87], [88], the authors propose a two-stage encoding process, where in the first stage data is encoded using a $[n, k]$ MDS code, and in the second stage $(n' - n)$ additional parities are generated to form a codeword from a $[n', k]$ MDS code. This process can be seen as a special case of convertible codes, i.e. an $(n, k; n', k)$ convertible code. In [89], the authors propose a distributed storage system which alternates between two specific erasure codes in response to variations in workload. In [90], the authors propose a scheme for changing the parameters of an erasure code in the context of coded matrix multiplication.

In [91], which appeared after the publication of the conference version of this paper [1], the authors propose a code construction for improving the efficiency of conversion. This construction performs conversion by acting on initial codewords that are encoded differently, i.e. a different $(k^I \times n^I)$ generator matrix is used for each initial codeword. The focus of Wu et al. [91] is on a practical code construction for a specific parameter regime and they do not investigate theoretical modeling and fundamental limits. All the lower bounds derived in our work continue to hold even if each codeword is encoded differently. The approach of using multiple different initial codes has the advantage of simplifying the code construction: a final MDS code $\mathcal{C}^F$ is chosen first, and then the encoding of each initial codeword is chosen to fit $\mathcal{C}^F$. However, such an approach has several disadvantages. First, conversion can only happen among specific groups of initial codewords, making the conversion process more rigid as codewords cannot be freely chosen. Second, this approach increases the overhead of codeword management, as the system needs to keep track of the code of each codeword. Third, it only considers one specific known value for the final parameters (n^F, k^F). On the other hand, the framework of convertible codes that we propose allows one to choose any set of initial codewords for conversion (since they all use the same code), is independent of data placement, and the proposed code constructions support access-optimal conversion for any (n^F, k^F) in a set of possible final parameter values.

B. Background

In this subsection we introduce some basic definitions and notation related to linear codes. Let $\mathbb{F}_q$ be a finite field of size q. An $[n, k]$ linear code $\mathcal{C}$ over $\mathbb{F}_q$ is a k-dimensional subspace $\mathcal{C} \subseteq \mathbb{F}_q^n$. Here, n is called the length of the code, and k is called the dimension of the code. A *generator matrix* of an $[n, k]$ linear code $\mathcal{C}$ over $\mathbb{F}_q$ is a $k \times n$ matrix $\mathbf{G}$ over $\mathbb{F}_q$ such that the rows of $\mathbf{G}$ form a basis of the subspace $\mathcal{C}$. A $k \times n$ generator matrix $\mathbf{G}$ is said to be *systematic* if it has the form $\mathbf{G} = [\mathbf{I} \mid \mathbf{P}]$, where $\mathbf{I}$ is the $k \times k$ identity matrix and $\mathbf{P}$ is a $k \times (n - k)$ matrix. Even though the generator matrix of a code $\mathcal{C}$ is not unique, we will sometimes associate a code $\mathcal{C}$ to a specific generator matrix $\mathbf{G}$, which will be clear from context. The encoding of a message $\mathbf{m} \in \mathbb{F}_q^k$ under an $[n, k]$ code $\mathcal{C}$ with generator matrix $\mathbf{G}$ is denoted $\mathcal{C}(\mathbf{m}) = \mathbf{m}^T\mathbf{G}$.

Let $[n]$ denote the set $\{1, 2, \ldots, n\}$ for $n \geq 1$, and the empty set for $n \leq 0$. A linear code $\mathcal{C}$ is *maximum distance separable* (MDS) if the minimum distance of the code is the maximum possible:

$$\text{min-dist}(\mathcal{C}) = \min_{c \neq c' \in \mathcal{C}} |\{i \in [n] : c_i \neq c'_i\}| = n - k + 1,$$

where $c_i \in \mathbb{F}_q$ denotes the i-th coordinate of c. Equivalently, a linear code $\mathcal{C}$ is MDS if and only if every $k \times k$ submatrix of its generator matrix $\mathbf{G}$ is non-singular [92].

A matrix M is said to be *superregular* if every square submatrix of M is nonsingular.[2] The following property is a key property that will be used in this paper.

[2]This definition of superregularity is stronger than the definition introduced in [93] in the context of convolutional codes.

Proposition 2 ([92]): Let $\mathcal{C}$ be an $[n, k]$ code with generator matrix $G = [I|P]$. Then $\mathcal{C}$ is MDS if and only if P is superregular.

Let $\mathbf{v} \in \mathbb{F}_q^n$ be a vector. We interpret vectors as column vectors by convention. We denote the transpose of a vector (or matrix) as $\mathbf{v}^T$. Given a set of coordinates $\mathcal{I} \subseteq [n]$, we denote the projection of $\mathbf{v}$ to the coordinates in $\mathcal{I}$ as $\mathbf{v}|_{\mathcal{I}} \in \mathbb{F}_q^{|\mathcal{I}|}$. For a set of vectors $\mathcal{V}$ we define $\operatorname{proj}_{\mathcal{I}}(\mathcal{V}) = \{\mathbf{v}|_{\mathcal{I}} \mid v \in \mathcal{V}\}$.

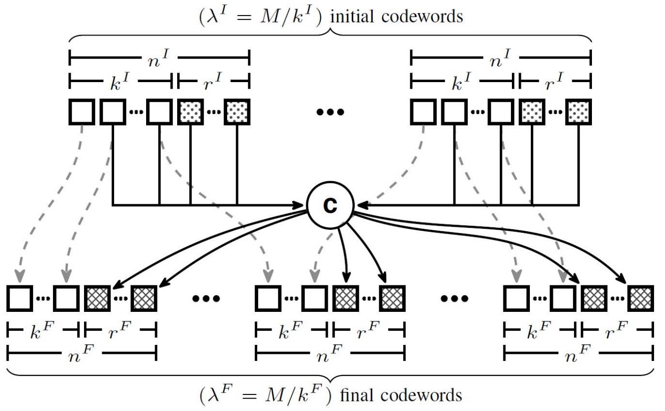
We use the following notation for submatrices: let M be a $n \times m$ matrix, the submatrix of M defined by row indices $\{i_1, \ldots, i_a\} \subseteq [n]$ and column indices $\{j_1, \ldots, j_b\} \subseteq [m]$ is denoted by $M[i_1, \ldots, i_a; j_1, \ldots, j_b]$. For conciseness, we use $*$ to denote all row or column indices, e.g., $M[*; j_1, \ldots, j_b]$ denotes the submatrix composed by columns $\{j_1, \ldots, j_b\}$, and $M[i_1, \ldots, i_a; *]$ denotes the submatrix composed by rows $\{i_1, \ldots, i_a\}$.

III. A Framework for Studying Code Conversions

In this section, we formally define the new framework for studying *code conversions* and introduce *convertible codes.* While we use the notation of linear codes introduced in Section II-B, the framework introduced in this section can be applied to arbitrary (not necessarily linear) codes. Suppose one wants to convert data that is already encoded using an $[n^I, k^I]$ initial code $\mathcal{C}^I$ into data encoded using an $[n^F, k^F]$ final code $\mathcal{C}^F$ where both codes are over the same field $\mathbb{F}_q$. In the initial and final configurations, the system must store the same information, but encoded differently. In order to capture the changes in the dimension of the code during conversion, we consider $M = \operatorname{lcm}(k^I, k^F)$ number of "message" symbols (i.e., the data to be stored) over a finite field $\mathbb{F}_q$, denoted by $\mathbf{m} \in \mathbb{F}_q^M$. This corresponds to $\lambda^I = M/k^I$ codewords in the initial configuration and $\lambda^F = M/k^F$ codewords in the final configuration. Let $r^I = (n^I - k^I)$ and $r^F = (n^F - k^F)$.

Figure 3 shows the conversion process for general initial and final codes. We note that this *need for considering multiple codewords in order to capture the smallest instance of the problem* deviates from existing literature on the code repair (e.g., [22], [23], [47], [59]) and code locality (e.g., [61], [66], [76]), where a single codeword is sufficient to capture the problem.

Since there are multiple codewords, we first specify an *initial partition* $\mathcal{P}^I$ and a *final partition* $\mathcal{P}^F$ of the set $[M]$, which map the message symbols of $\mathbf{m}$ to their corresponding initial and final codewords. The initial partition $\mathcal{P}^I = \{P_1^I, \ldots, P_{\lambda^I}^I\}$ is composed of λ^I disjoint subsets of size $|P_i^I| = k^I$ ($i \in [\lambda^I]$), and the final partition $\mathcal{P}^F = \{P_1^F, \ldots, P_{\lambda^F}^F\}$ is composed of λ^F disjoint subsets of size $|P_j^F| = k^F$ ($j \in [\lambda^F]$). In the initial (respectively, final) configuration, the data indexed by each subset $P_i^I \in \mathcal{P}^I$ (respectively, $P_j^F \in \mathcal{P}^F$) is encoded using the code $\mathcal{C}^I$ (respectively, $\mathcal{C}^F$). The codewords $\{\mathcal{C}^I(\mathbf{m}|_{P_i^I}) \mid P_i^I \in \mathcal{P}^I\}$ are referred to as *initial codewords*, and the codewords $\{\mathcal{C}^F(\mathbf{m}|_{P_j^F}) \mid P_j^F \in \mathcal{P}^F\}$ are referred to as *final codewords.* The descriptions of the initial and final partitions and codes, along with the conversion procedure, define a convertible code. We now proceed to define conversions and convertible codes formally.



$(\lambda^I = M/k^I)$ initial codewords
n^I
k^I
r^I
C
k^F
r^F
n^F
$(\lambda^F = M/k^F)$ final codewords

Fig. 3. Conversion from $[n^I, k^I]$ initial code to $[n^F, k^F]$ final code. Each box denotes a symbol, and they are grouped into codewords. Dotted boxes denote retired symbols, and cross-hatched boxes denote new symbols. The **c** node denotes the location where new symbols are computed from the symbols read during conversion. Solid arrows denote a transfer of symbols (read or write) and dashed arrows denote unchanged symbols.

Definition 3 (Code Conversion): A *conversion* from an initial code $\mathcal{C}^I$ to a final code $\mathcal{C}^F$ with initial partition $\mathcal{P}^I$ and final partition $\mathcal{P}^F$ is a procedure, denoted by $T_{\mathcal{C}^I \to \mathcal{C}^F}$, that for any $\mathbf{m}$, takes the set of initial codewords $\{\mathcal{C}^I(\mathbf{m}|_{P_i^I}) \mid P_i^I \in \mathcal{P}^I\}$ as input, and outputs the corresponding set of final codewords $\{\mathcal{C}^F(\mathbf{m}|_{P_j^F}) \mid P_j^F \in \mathcal{P}^F\}$. ▶

Definition 4 (Convertible Code): An $(n^I, k^I; n^F, k^F)$ convertible code over $\mathbb{F}_q$ is defined by: (1) a pair of codes $(\mathcal{C}^I, \mathcal{C}^F)$ where $\mathcal{C}^I$ is an $[n^I, k^I]$ code over $\mathbb{F}_q$ and $\mathcal{C}^F$ is an $[n^F, k^F]$ code over $\mathbb{F}_q$; (2) a pair of partitions $\mathcal{P}^I, \mathcal{P}^F$ of $[M = \operatorname{lcm}(k^I, k^F)]$ such that each subset in $\mathcal{P}^I$ is of size k^I and each subset in $\mathcal{P}^F$ is of size k^F; and (3) a conversion procedure $T_{\mathcal{C}^I \to \mathcal{C}^F}$ that on input $\{\mathcal{C}^I(\mathbf{m}|_{P_i^I}) \mid P_i^I \in \mathcal{P}^I\}$ outputs $\{\mathcal{C}^F(\mathbf{m}|_{P_j^F}) \mid P_j^F \in \mathcal{P}^F\}$, for any $\mathbf{m} \in \mathbb{F}_q^M$. ▶

Typically, additional constraints would be imposed on $\mathcal{C}^I$ and $\mathcal{C}^F$, for example, decodability constraints such as requiring both codes to be MDS.

The cost of conversion is determined by the cost of the conversion procedure $T_{\mathcal{C}^I \to \mathcal{C}^F}$, as a function of the parameters $(n^I, k^I; n^F, k^F)$. Towards minimizing the overhead of the conversion, our general objective is to design codes $(\mathcal{C}^I, \mathcal{C}^F)$, partitions $(\mathcal{P}^I, \mathcal{P}^F)$ and conversion procedure $T_{\mathcal{C}^I \to \mathcal{C}^F}$ that satisfy Definition 4 and minimize the conversion cost for given parameters $(n^I, k^I; n^F, k^F)$, subject to desired decodability constraints on $\mathcal{C}^I$ and $\mathcal{C}^F$.

Depending on the relative importance of various resources in the cluster, one might be interested in optimizing the conversion with respect to various types of costs such as symbol access, computation (CPU), communication (network bandwidth), read/writes (disk IO), etc., or a combination of these costs. The general formulation of code conversions above provides a powerful framework to theoretically reason about convertible codes. In this paper, we focus on a specific cost model.

As a measure of cost, we consider the access cost of code conversion, which measures the number of symbols that are affected by the conversion.

TABLE II

NOTATION USED IN THE PAPER

$\square^I$	Related to initial code	$\square^F$	Related to final code	ς	Number of initial codewords (merge regime)
$n^\diamondsuit$	Code length, number of symbols	$k^\diamondsuit$	Code dimension, number of message symbols	$\lambda^\diamondsuit$	Number of codewords
$\mathcal{C}^\diamondsuit$	Code	$\mathcal{P}^\diamondsuit$	Partition of $[k^\diamondsuit]$	$\mathbf{G}^\diamondsuit$	Generator matrix of $\mathcal{C}^\diamondsuit$
$\mathbf{P}^\diamondsuit$	Parity matrix of $\mathcal{C}^\diamondsuit$	$\mathbf{m}$	Message	$\mathcal{S}_i^I$	Encoding vectors (initial codeword i)
$\mathcal{S}^F$	Encoding vectors (final codeword)	$\mathcal{U}_i$	Unchanged vectors ($= \mathcal{S}_i^I \cap \mathcal{S}^F$)	$\mathcal{D}_i$	Read access set (initial codeword i)
$\mathcal{A}_i$	Accessed vectors (initial codeword i)	$\mathcal{N}$	New encoding vectors ($= \mathcal{S}^F \setminus \mathcal{S}^I$)		

Definition 5 (Access Cost): The *read access cost* of a conversion procedure is defined as the total number of symbols read during the procedure. Similarly, the *write access cost* of a conversion procedure is the total number of symbols written during the procedure. The *access cost* of a conversion procedure is the sum of its read and write access costs. The access cost of a convertible code is the access cost of its conversion procedure.

Each symbol read from the initial codewords requires one symbol access and each symbol written to the final codewords requires one symbol access. Therefore, minimizing access cost amounts to minimizing the sum of the number of symbols read from the initial codewords and the number of symbols written to the final codewords.[3] Keeping this number small makes code conversion less disruptive and allows the unaffected symbols to remain available for normal operation. Furthermore, reducing the number of accesses also reduces the amount of computation and communication required in contrast to the default approach.

In order to understand the necessary access cost of conversion, we classify symbols into three categories: (1) *unchanged symbols*, which refers to symbols in the initial codewords that remain *as is* in the final codewords; (2) *retired symbols*, which refers to the remaining symbols of the initial codewords that are discarded; and (3) *new symbols*, which refers the symbols in the final stripes which are not unchanged (and therefore must be written during conversion). For example, in Figure 3, unchanged symbols are unshaded, retired symbols in the initial codewords are dotted, and new symbols in the final codewords are cross-hatched.

Having unchanged symbols has many practical benefits, because when conversion is implemented, such symbols can stay in the same location and only their corresponding metadata needs to be updated. We introduce the following definition to capture codes that maximize the number of such symbols.

Definition 6 (Stable Convertible Code): An $(n^I, k^I; n^F, k^F)$ MDS convertible code is said to be *stable* if it uses the maximum number of unchanged symbols over all $(n^I, k^I; n^F, k^F)$ MDS convertible codes. ▶

We will see in Section IV, that stable convertible codes play an important role in minimizing access cost.

The convertible codes framework defined in this work is flexible and allows for the initial and final codes to have any parameters and be of any kind. However, as a step towards a fundamental theoretical understanding of the access cost of conversions, in this paper we focus in a particular subclass of conversions that we term *merge regime*. The merge regime correspond to code conversions where multiple initial codewords are combined into a single codeword. In other words, the merge regime consists of convertible codes where $k^F = \varsigma k^I$, for some integer $\varsigma \geq 2$, and n^I, n^F are arbitrary. In addition to this, in this paper we focus exclusively on codes that are both linear and MDS.

In particular, our goal is to find linear MDS codes which can achieve conversion with the minimum possible access cost.

Definition 7 (Access-Optimal): A linear MDS $(n^I, k^I; n^F, k^F)$ convertible code is said to be *access-optimal* if and only if it attains the minimum access cost over all linear MDS $(n^I, k^I; n^F, k^F)$ convertible codes. ▶

We will study the precise cost of access-optimal convertible codes in the merge regime in Section IV.

In practice, the final parameters (n^F, k^F) might not be known at the time of code construction because they might depend on future failure rates. To address this, we also consider designing codes which have the ability to be converted to multiple final codes of different length and dimension with optimal access cost. This way, instead of having to decide (n^F, k^F) in advance, the user can specify a subset $S \subseteq (\mathbb{N}\times\mathbb{N})$ of possible values for the pair (n^F, k^F) and construct an initial code with the ability to be converted to an $[n^F, k^F]$ final code for *any* $(n^F, k^F) \in S$. At the time of conversion, the user simply chooses the desired pair from S and converts. We introduce the following definition to help describe such codes.

Definition 8 (Access-Optimally Convertible): A linear $[n^I, k^I]$ MDS code $\mathcal{C}^I$ is said to be (n^F, k^F)-access-optimally convertible if and only if it is the initial code of an access-optimal $(n^I, k^I; n^F, k^F)$ convertible code. ▶

A. Notation for Linear Convertible Codes in the Merge Regime

In this paper, we focus exclusively on convertible codes in the merge regime where $\mathcal{C}^I$ and $\mathcal{C}^F$ are linear. To this end, we introduce some notation for describing and analyzing this class of codes. Table II summarizes the most important notation used for easy reference.

First, we make some observations that help simplify the notation for the merge regime. In the merge regime $k^F = \varsigma k^I$ and, as a consequence, the length of the message $\mathbf{m}$ will be $M = \text{lcm}(k^I, k^F) = k^F$. Furthermore, the number of initial codewords will be $\lambda^I = \varsigma$ and the number of final codewords

[3]Readers who are familiar with the literature on regenerating codes might observe that convertible codes optimizing for the access cost are "scalar" codes as opposed to being "vector" codes.

will be $\lambda^F = 1$. Therefore, we need not specify which final codeword we refer to, as there is only one.

Let $\Diamond \in \{I, F\}$. The generator matrix of $\mathcal{C}^\Diamond$ is a $(k^\Diamond \times n^\Diamond)$ matrix $\mathbf{G}^\Diamond = [\mathbf{g}_1^\Diamond \cdots \mathbf{g}_{n^\Diamond}^\Diamond]$, where $\mathbf{g}_j^\Diamond \in \mathbb{F}_q^{k^\Diamond}$ $(j \in [n^\Diamond])$ denotes the j-th *encoding vector* of $\mathcal{C}^\Diamond$. Consequently, the j-th symbol of the i-th codeword corresponds to $(\mathbf{m}|_{P_i^\Diamond})^T \mathbf{g}_j^\Diamond$.

In order to analyse linear convertible codes, we also view each code symbol in relation to the whole message $\mathbf{m}$. Accordingly, we view the j-th symbol of the i-th initial codeword as $\mathbf{m}^T \tilde{\mathbf{g}}_{i,j}^I$, where the encoding vector $\tilde{\mathbf{g}}_{i,j}^I \in \mathbb{F}_q^M$ is defined to be equal to $\mathbf{g}_j^I$ for coordinates in P_i^I, i.e. $\tilde{\mathbf{g}}_{i,j}^I|_{P_i^I} = \mathbf{g}_j^I$, and equal to 0 everywhere outside of P_i^I. Note that $\mathbf{m}^T \tilde{\mathbf{g}}_{i,j}^I = (\mathbf{m}|_{P_i^I})^T \mathbf{g}_j^I$ for all $i \in [\varsigma]$ and $j \in [n^I]$. In general, we will refer to a code symbol and its corresponding encoding vector interchangeably.

Let $\mathcal{S}_i^I = \{\tilde{\mathbf{g}}_{i,j}^I \mid j \in [n^I]\}$ denote the encoding vectors of initial codeword $i \in [\varsigma]$, let $\mathcal{S}^I = \bigcup_{i \in [\varsigma]} \mathcal{S}_i^I$, and let $\mathcal{S}^F = \{\mathbf{g}_j^F \mid j \in [n^F]\}$. Define $\mathcal{U}_i = (\mathcal{S}_i^I \cap \mathcal{S}^F)$ denoting the unchanged symbols of initial codeword i, and let $\mathcal{U} = (\mathcal{S}^I \cap \mathcal{S}^F)$ denote all unchanged vectors. We define the *read access set* of a convertible code as a set of tuples $\mathcal{D} \in [\lambda^I] \times [n^I]$, where $(i, j) \in \mathcal{D}$ corresponds to the j-th symbol of initial codeword i. Furthermore, we use $\mathcal{D}_i = \{j \mid (i, j) \in \mathcal{D}\}$, $\forall i \in [\lambda^I]$ to denote the symbols read from initial codeword i. Note that the read access cost is given by $|\mathcal{D}|$. Let $\mathcal{A}_i = \{\tilde{\mathbf{g}}_{i,j}^I \mid j \in \mathcal{D}_i\}$ denote the encoding vectors of the symbols from initial codeword $i \in [\lambda^I]$ that are part of the read access set $\mathcal{D}$, and define $\mathcal{A} = \{\tilde{\mathbf{g}}_{i,j}^I \mid (i, j) \in \mathcal{D}\}$ as the set of all encoding vectors of the symbols in the read access set. Finally, let $\mathcal{N} = (\mathcal{S}^F \setminus \mathcal{S}^I)$ denote the new vectors. Notice that it must hold that $\mathcal{N} \subseteq \operatorname{span}(\mathcal{A})$, since the new vectors are obtained as linear combinations of the encoding vectors of the symbols in the read access set.

B. The Case of $k^I = k^F$

Before analyzing the conversion in the merge regime, we briefly study the exceptional conversion case where $k^I = k^F$. Conversion with parameters $k^I = k^F$ is not considered as part of the merge regime because it does not have the same behavior. However, we analyze this case here for completeness. Observe that in the case where $n^I \geq n^F$, conversion for any MDS code can be carried out with zero access cost by simply retiring any $(n^I - n^F)$ symbols. In the complementary case where $n^I < n^F$, it is necessary to access at least k^I symbols and write at least $(n^F - n^I)$ symbols (i.e. it is not possible to beat the default approach). This is apparent from the fact that in an $[n, k]$ MDS code, any subset of $k-1$ symbols gives no information about any one of the remaining symbols. Therefore, in the remainder of this paper, we consider $\varsigma \geq 2$.

IV. Lower Bounds on the Access Cost of Convertible Codes in the Merge Regime

In this paper, we focus on studying the merge regime. Recall, from Section III, that the merge regime corresponds to conversion where multiple codewords are combined into a single codeword (i.e. $k^F = \varsigma k^I$ for an integer $\varsigma \geq 2$). This implies that $M = k^F$, $\lambda^I = \varsigma$, and $\lambda^F = 1$.

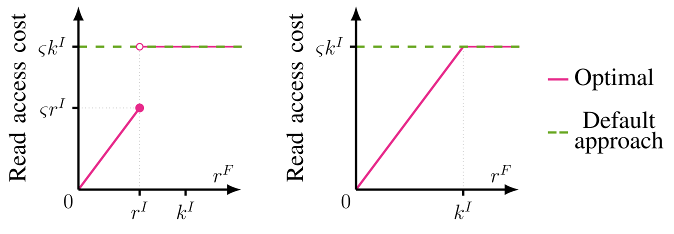


Fig. 4. Comparison of the read access cost of the optimal conversion of a $(n^I, k^I; n^F, k^F = \varsigma k^I)$ convertible code and the default approach for a variable value of r^F (x-axis) when $r^I < k^I$ (left side) and $r^I \geq k^I$ (right side). When $r^F < \min\{k^I, r^I\}$, optimal conversion achieves lower cost than the default approach, and when $r^F > \min\{k^I, r^I\}$, the default approach is (trivially) optimal. The optimal write access cost in the merge regime is always r^F.

TABLE III
Access Cost Savings for Different Example Parameters

$[n^I, k^I] \Rightarrow [n^F, k^F]$	Optimal access cost	Default approach	Cost reduction
$[14, 10] \Rightarrow [22, 20]$	6	22	72.7%
$[9, 6] \Rightarrow [14, 12]$	6	14	57.1%
$[5, 3] \Rightarrow [15, 9]$	9	15	40.0%
$[9, 5] \Rightarrow [14, 10]$	12	14	14.3%
$[6, 4] \Rightarrow [11, 8]$	11	11	0.0%

In this section, we present lower bounds on the access cost of linear MDS convertible codes in the merge regime. Our main result is summarized by the following theorem, which will be proved at the end of this section.

Theorem 9: For all linear MDS $(n^I, k^I; n^F, k^F = \varsigma k^I)$ convertible codes, the read access cost of conversion is at least $\varsigma \min\{k^I, r^F\}$ and the write access cost is at least r^F. Furthermore, if $r^I < r^F$, the read access cost of conversion is at least ςk^I.

As we will show in Section V, this lower bound is achievable and it therefore corresponds to the optimal access cost in the merge regime. Figure 4 shows a plot comparing the optimal access cost against the access cost of the default approach for different parameter values, and Table III shows these costs for some concrete conversion examples.

We break down the proof of this result into four steps:

1) We show that in the merge regime, all possible pairs of partitions $\mathcal{P}^I$ and $\mathcal{P}^F$ partitions are equivalent up to relabeling, and hence do not need to be specified (Lemma 10).
2) An upper bound on the maximum number of unchanged symbols is proved. As described in Definition 6, convertible codes that meet this bound are called stable (Lemma 11).
3) Lower bounds on the access cost of linear MDS convertible codes are proved under the added restriction that the codes are stable (Lemmas 12 and 13 and Theorem 14).
4) The stability restriction is removed, by showing that non-stable linear MDS convertible codes necessarily incur higher access cost, and hence it suffices to consider only stable MDS convertible codes (Lemma 16 and Theorem 9).

In general, partitions need to be specified since they indicate how message symbols from the initial codewords are mapped into the final codewords. However in the merge regime, the choice of the partitions are equivalent, and hence are inconsequential as shown below.

Lemma 10: For every $(n^I, k^I; n^F, k^F = \varsigma k^I)$ convertible code, all possible pairs of initial and final partitions $(\mathcal{P}^I, \mathcal{P}^F)$ are equivalent up to relabeling of nodes.

Proof: We have that $k^I \mid k^F$. Thus $\lambda^F = (M/k^F) = 1$ and $\mathcal{P}^F = \{[M]\}$ always holds. Because of this, all data will be mapped to the same final codeword, regardless of the initial partition. Therefore, for any two partitions $\mathcal{P}^I$ and $\mathcal{P}^{I'}$, there exists some permutation σ of $[\varsigma k^I]$ such that $\mathcal{P}^{I'} = \{\sigma(P) \mid P \in \mathcal{P}^I\}$, i.e., different partitions differ only on the way nodes are labeled. ■

Since one of the terms in access cost is the number of new symbols, a natural way to reduce access cost is to maximize the number of unchanged symbols. However, there is a limit on the number of symbols that can remain unchanged which is characterized below.

Lemma 11: In an MDS $(n^I, k^I; n^F, k^F = \varsigma k^I)$ convertible code, there can be at most k^I unchanged symbols from each initial codeword. ■

Proof: By the MDS property of $\mathcal{C}^I$ every subset of $k^I + 1$ symbols is linearly dependent. Hence, there can be at most k^I unchanged symbols from each initial codeword for $\mathcal{C}^F$ to be MDS. In other words, $|\mathcal{U}_i| \leq k^I$ for all $i \in [\varsigma]$. ■

This implies that there are at most ςk^I unchanged symbols and at least r^F new symbols in total. Thus, the number of symbols that need to be written in a stable code is at least r^F.

Now, we focus on bounding the total number of symbols read, that is, the size of the read access sets. The general strategy we use to obtain bounds on the size of read access sets is to consider a specially chosen set of k^F encoding vectors from the final codeword, which by the MDS property of the final code is linearly independent. We then use the fact that final codewords are the result of conversion to identify the encoding vectors in each initial codeword that span the selected final encoding vectors. The MDS property of the initial code and the fact that different initial codewords contain different information will allow us to derive a lower bound on the number of read symbols in each initial codeword.

Intuitively, having more new symbols means that more symbols have to be read in order to construct them, resulting in higher access cost. With this intuition in mind, we first focus on stable convertible codes, which minimize the number of new symbols (Definition 6). We first prove lower bounds on the access cost of stable linear MDS convertible codes, and then show that the minimum access cost of conversion in MDS codes without this stability property can only be higher. The first lower bound on the size of each $\mathcal{D}_i$ $(i \in [\varsigma])$ is given by the interaction between new symbols and the MDS property.

Lemma 12: For every linear stable MDS $(n^I, k^I; n^F, k^F = \varsigma k^I)$ convertible code, the read access set $\mathcal{D}_i$ from each initial codeword $i \in [\varsigma]$ satisfies $|\mathcal{D}_i| \geq \min\{k^I, r^F\}$.

Proof: For convenience, readers can recall the notation from Table II. By the MDS property, every subset $\mathcal{V} \subseteq \mathcal{S}^F$ of size at most $k^F = \varsigma k^I$ is linearly independent. For any initial codeword $i \in [\varsigma]$, take the set of all unchanged encoding vectors from other codewords $\cup_{\ell \neq i} \mathcal{U}_\ell$, and additionally pick any subset of new encoding vectors $\mathcal{W} \subseteq \mathcal{N}$ of size $|\mathcal{W}| = \min\{k^I, r^F\}$. The following holds for set $\mathcal{V} = (\cup_{\ell \neq i} \mathcal{U}_\ell \cup \mathcal{W})$:

$$\mathcal{V} \subseteq \mathcal{S}^F \text{ and } |\mathcal{V}| = (\varsigma - 1)k^I + \min\{k^I, r^F\} \leq k^F.$$

Therefore, all the encoding vectors in $\mathcal{V}$ are linearly independent.

Notice that the encoding vectors in $(\mathcal{V} \setminus \mathcal{W})$ contain no information about initial codeword i and complete information about every other initial codeword $\ell \neq i$. Therefore, the information about initial codeword i in each encoding vector in $\mathcal{W}$ has to be linearly independent since, otherwise, $\mathcal{V}$ could not be linearly independent. Formally, it must be the case that $\mathcal{W}_i = \text{proj}_{P_i^I}(\mathcal{W})$ has rank equal to $\min\{k^I, r^F\}$ (recall that P_i^I is the set of symbols corresponding to initial codeword i). However, by definition, the subset $\mathcal{W}_i$ must be contained in the span of $\mathcal{A}_i$. Therefore, the rank of $\mathcal{A}_i$ is at least that of $\mathcal{W}_i$, which implies that $|\mathcal{D}_i| \geq \min\{k^I, r^F\}$. ■

We next show that when the number of new symbols r^F is greater than r^I in a MDS stable convertible code in the merge regime, then the default approach is optimal in terms of access cost.

Lemma 13: For every linear stable MDS $(n^I, k^I; n^F, k^F = \varsigma k^I)$ convertible code, if $r^I < r^F$ then the read access set $\mathcal{D}_i$ from each initial codeword $i \in [\varsigma]$ satisfies $|\mathcal{D}_i| \geq k^I$.

Proof: When $r^F \geq k^I$, this lemma is equivalent to Lemma 12, so assume $r^I < r^F < k^I$. From the proof of Lemma 12, for every initial codeword $i \in [\varsigma]$ it holds that $|\mathcal{D}_i| \geq r^F$. Since $r^F > r^I$, this implies that $\mathcal{D}_i$ must contain at least one index of an unchanged encoding vector.

Choose a subset of at most $k^F = \varsigma k^I$ encoding vectors from $\mathcal{S}^F$, which must be linearly independent by the MDS property. In this subset, include all the unchanged encoding vectors from the other initial codewords, $\cup_{l \neq i} \mathcal{U}_l$. Then, choose all the unchanged encoding vectors from initial codeword i that are accessed during conversion, $\mathcal{W}_1 = (\mathcal{A}_i \cap \mathcal{U}_i)$. For the remaining vectors (if any), choose an arbitrary subset of new encoding vectors, $\mathcal{W}_2 \subseteq \mathcal{N}$, such that:

$$|\mathcal{W}_2| = \min\{k^I - |\mathcal{W}_1|, r^F\}. \quad (1)$$

It is easy to check that the subset $\mathcal{V} = (\cup_{l \neq i} \mathcal{U}_l \cup \mathcal{W}_1 \cup \mathcal{W}_2)$ is of size at most $k^F = \varsigma k^I$, and therefore it is linearly independent. This choice of $\mathcal{V}$ follows from the idea that the information contributed by $\mathcal{W}_1$ to the new encoding vectors is already present in the unchanged encoding vectors, which will be at odds with the linear independence of $\mathcal{V}$.

Since the elements of $\mathcal{W}_1$ and $\mathcal{W}_2$ are the only encoding vectors in $\mathcal{V}$ that contain information from initial codeword i, it must be the case that $\widetilde{\mathcal{W}} = (\text{proj}_{P_i^I}(\mathcal{W}_1) \cup \text{proj}_{P_i^I}(\mathcal{W}_2))$ has rank $(|\mathcal{W}_1| + |\mathcal{W}_2|)$. Moreover, $\widetilde{\mathcal{W}}$ is contained in the span of $\mathcal{A}_i$ by definition, so it holds that:

$$|\mathcal{D}_i| \geq |\mathcal{W}_1| + |\mathcal{W}_2|. \quad (2)$$

From Equation 1, there are two cases:

Case 1: $(k^I - |\mathcal{W}_1|) \leq r^F$. Then $|\mathcal{W}_2| = (k^I - |\mathcal{W}_1|)$ and by Equation 2 it holds that $|\mathcal{D}_i| \geq (|\mathcal{W}_1| + |\mathcal{W}_2|) = k^I$.

Case 2: $(k^I - |\mathcal{W}_1|) > r^F$. Then $|\mathcal{W}_2| = r^F$ and by Equation 2 it holds that:

$$|\mathcal{D}_i| \geq |\mathcal{W}_1| + r^F. \tag{3}$$

Notice that there are only r^I retired (i.e. not unchanged) encoding vectors in codeword i. Since every accessed encoding vector is either in $\mathcal{W}_1$ or is a retired encoding vector, it holds that:

$$|\mathcal{D}_i| \leq |\mathcal{W}_1| + r^I. \tag{4}$$

By combining Equation 3 and Equation 4, we arrive at the contradiction $r^F \leq r^I$, which occurs because there are not enough retired symbols in the initial codeword i to ensure that the final code has the MDS property. Therefore, case 1 must always hold, and $|\mathcal{D}_i| \geq k^I$. ■

Combining the above results leads to the following theorem on the lower bound of read access set size of linear stable MDS convertible codes.

Theorem 14: For all stable linear MDS $(n^I, k^I; n^F, k^F = \varsigma k^I)$ convertible codes with read access set $\mathcal{D}$, it holds that $|\mathcal{D}| \geq \varsigma \min\{k^I, r^F\}$. Furthermore, if $r^I < r^F$, then $|\mathcal{D}| \geq k^F$.

Proof: Follows directly from Lemma 12 and Lemma 13. ■

We next show that this lower bound generally applies even for non-stable convertible codes by proving that increasing the number of new symbols from the minimum possible does not decrease the lower bound on the size of the read access set $\mathcal{D}$.

Lemma 15: The lower bounds on the size of the read access set from Lemma 14 hold for **all** linear MDS $(n^I, k^I; n^F, k^F = \varsigma k^I)$ convertible codes.

Proof: We show that, even for non-stable convertible codes, that is, when there are more than r^F new symbols, the bounds on the read access set $\mathcal{D}$ from Theorem 14 still hold.

Case 1: $r^I \geq r^F$. Let $i \in [\varsigma]$ be an arbitrary initial codeword. We lower bound the size of $\mathcal{D}_i$ by invoking the MDS property on a subset $\mathcal{V} \subseteq \mathcal{S}^F$ of size $|\mathcal{V}| = \varsigma k^I$ that minimizes the size of the intersection $|\mathcal{V} \cap \mathcal{U}_i|$. There are exactly r^F encoding vectors in $(\mathcal{S}^F \setminus \mathcal{V})$, so the minimum size of the intersection $|\mathcal{V} \cap \mathcal{U}_i|$ is $\max\{|\mathcal{U}_i| - r^F, 0\}$. Clearly, the subset $\mathrm{proj}_{P_i^I}(\mathcal{V})$ has rank k^I due to the MDS property. Therefore, it holds that $|\mathcal{D}_i| + \max\{|\mathcal{U}_i| - r^F, 0\} \geq k^I$. By reordering, the following is obtained:

$$|\mathcal{D}_i| \geq k^I - \max\{|\mathcal{U}_i| - r^F, 0\} \geq \min\{r^F, k^I\},$$

which means that the bound on $\mathcal{D}_i$ established in Lemma 12 continues to hold for non-stable codes.

Case 2: $r^I < r^F$. Let $i \in [\varsigma]$ be an arbitrary initial codeword, let $\mathcal{W}_1 = (\mathcal{A}_i \cap \mathcal{U}_i)$ be the unchanged encoding vectors that are accessed during conversion, and let $\mathcal{W}_2 = (\mathcal{U}_i \setminus \mathcal{W}_1)$ be the unchanged encoding vectors that are *not* accessed during conversion. Consider the subset $\mathcal{V} \subseteq \mathcal{S}^F$ of $|\mathcal{V}| = k^F$ encoding vectors from the final codeword such that $\mathcal{V} \supseteq \mathcal{W}_1$ and the size of the intersection $\mathcal{W}_3 = (\mathcal{V} \cap \mathcal{W}_2)$ is minimized. Since $\mathcal{V}$ may exclude at most r^F encoding vectors from the final codeword, it holds that:

$$|\mathcal{W}_3| = \max\{0, |\mathcal{W}_2| - r^F\}. \tag{5}$$

By the MDS property, $\mathcal{V}$ is a linearly independent set of encoding vectors of size k^F, and thus, must contain all the information to recover the contents of every initial codeword, and in particular, initial codeword i. Since all the information in $\mathcal{V}$ about codeword i is in either $\mathcal{W}_3$ or the accessed encoding vectors, it must hold that:

$$|\mathcal{D}_i| + |\mathcal{W}_3| \geq k^I. \tag{6}$$

From Equation 5, there are two cases:

Subcase 2.1: $(|\mathcal{W}_2| - r^F) \leq 0$. Then $|\mathcal{W}_3| = 0$, and by Equation 6 it holds that $|\mathcal{D}_i| \geq k^I$, which matches the bound of Lemma 13.

Subcase 2.2: $(|\mathcal{W}_2| - r^F) > 0$. Then $|\mathcal{W}_3| = (|\mathcal{W}_2| - r^F)$, and by Equation 6 it holds that:

$$|\mathcal{D}_i| + |\mathcal{W}_2| - r^F \geq k^I. \tag{7}$$

The initial codeword i has $(k^I + r^I)$ symbols. By the principle of inclusion-exclusion we have that:

$$|\mathcal{D}_i| + |\mathcal{U}_i| - |\mathcal{W}_1| \leq k^I + r^I. \tag{8}$$

By using Equation 7, Equation 8 and the fact that $|\mathcal{W}_2| = (|\mathcal{U}_i| - |\mathcal{W}_1|)$, we conclude that $r^I \geq r^F$, which is a contradiction and means that subcase 2.1 always holds in this case. ■

The above result, along with the fact that the lower bound in Theorem 14 is achievable (as will be shown in Section V), implies that all access-optimal linear MDS convertible codes in the merge regime are stable.

Lemma 16: All access-optimal linear MDS $(n^I, k^I; n^F, k^F = \varsigma k^I)$ convertible codes are stable.

Proof: Lemma 15 shows that the lower bound on the read access set $\mathcal{D}$ for stable linear MDS convertible codes continues to hold in the non-stable case. Furthermore, this bound is achievable by stable linear MDS convertible codes in the merge regime (as will be shown in Section V). The number of new blocks written during conversion under stable MDS convertible codes is r^F. On the other hand, the number of new symbols under a non-stable convertible code is strictly greater than r^F. Thus, the overall access cost of a non-stable MDS $(n^I, k^I; n^F, k^F = \varsigma k^I)$ convertible code is strictly greater than the access cost of an access-optimal $(n^I, k^I; n^F, k^F = \varsigma k^I)$ convertible code. ■

Thus, for MDS convertible codes in the merge regime, it suffices to focus only on stable codes. Combining all the results above, leads to the main theorem presented at the beginning of this section.

Proof of Theorem 9: Follows from Theorem 14 and Lemmas 15 and 16, and the fact that at least r^F new symbols must be written. ■

Next, in Section V we show that the lower bound of Theorem 9 is achievable for all parameters. Thus, Theorem 9 implies that it is possible to perform conversion of MDS convertible codes in the merge regime with significantly less access cost than the default approach if and only if $r^F \leq r^I$ and $r^F < k^I$.

V. Achievability: Explicit Access-Optimal Convertible Codes in the Merge Regime

In this section, we present an explicit construction of access-optimal MDS convertible codes for all parameters in the merge regime. In other words, we present a construction that matches the access cost lower bound presented in Section IV. In Section V-A, we present the construction of the generator matrices for the initial and final code. Then, in Section V-B, we describe sufficient conditions for optimality and show that this construction satisfies these conditions and thus yields access-optimal convertible codes. Our constructions in this and the following section work over any finite field of sufficient size (which we explicitly specify), but for the sake of illustration we use prime fields in our examples.

A. Explicit Construction of Generator Matrices

Recall that, in the merge regime, $k^F = \varsigma k^I$, for an integer $\varsigma \geq 2$, while $n^I > k^I$ and $n^F > k^F$ are arbitrary. Also, recall that $r^I = (n^I - k^I)$ and $r^F = (n^F - k^F)$. Notice that when $r^I < r^F$ or $k^I \leq r^F$, constructing an access-optimal convertible code is trivial, since the default approach to conversion is optimal. Thus, assume $r^F \leq \min\{r^I, k^I\}$.

Let $\mathbb{F}_q$ be a finite field of size $q = p^D$, where p is any prime (in particular, we can have $p = 2$, i.e. a binary field) and the degree D is determined by a function of the convertible code parameters (discussed later in this subsection). The degree D required by this construction is $\mathcal{O}((\max\{n^I, n^F\})^3)$, that is, the field size requirement is exponential in the length of the code. Let θ be a primitive element of $\mathbb{F}_q$. Let $\mathbf{G}^I = [\mathbf{I}|\mathbf{P}^I]$ and $\mathbf{G}^F = [\mathbf{I}|\mathbf{P}^F]$ be systematic generator matrices of $\mathcal{C}^I$ and $\mathcal{C}^F$ over $\mathbb{F}_q$, where $\mathbf{P}^I$ is a $k^I \times r^I$ matrix and $\mathbf{P}^F$ is a $k^F \times r^F$ matrix.

Define entry (i, j) of $\mathbf{P}^I \in \mathbb{F}_q^{k^I \times r^I}$ as $\theta^{(i-1)(j-1)}$, where (i, j) ranges over $[k^I] \times [r^I]$. Entry (i, j) of $\mathbf{P}^F \in \mathbb{F}_q^{k^F \times r^F}$ is defined identically as $\theta^{(i-1)(j-1)}$, where (i, j) ranges over $[k^F] \times [r^F]$. That is, $\mathbf{P}^I$ and $\mathbf{P}^F$ are as follows:

$$\mathbf{P}^I = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \theta & \theta^2 & \cdots & \theta^{(r^I-1)} \\ 1 & \theta^2 & \theta^4 & \cdots & \theta^{2(r^I-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \theta^{(k^I-1)} & \theta^{2(k^I-1)} & \cdots & \theta^{(k^I-1)(r^I-1)} \end{bmatrix},$$

$$\mathbf{P}^F = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \theta & \theta^2 & \cdots & \theta^{(r^F-1)} \\ 1 & \theta^2 & \theta^4 & \cdots & \theta^{2(r^F-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \theta^{(k^F-1)} & \theta^{2(k^F-1)} & \cdots & \theta^{(k^F-1)(r^F-1)} \end{bmatrix}.$$

Notice that this construction is stable, because it is access-optimal (recall Lemma 16). The unchanged symbols of the initial code are exactly the systematic symbols.

B. Proof of Optimality

Recall from Proposition 2, that if the constructed code is to be MDS, then both $\mathbf{P}^I$ and $\mathbf{P}^F$ need to be superregular (every square submatrix of them is invertible). In addition, to be access-optimal during conversion in the non-trivial case, the new symbols (corresponding to the columns of $\mathbf{P}^F$) have to be such that they can be generated by accessing r^F symbols from the initial codewords (corresponding to columns of $\mathbf{G}^I$).

During conversion, the encoding vectors of symbols from the initial codewords are represented as ςk^I-dimensional vectors, where each initial codeword occupies a disjoint subset of k^I coordinates. To capture this property, we introduce the following definition.

Definition 17 (t-Column Block-Constructible): We will say that an $n \times m_1$ matrix M_1 is *t-column constructible* from an $n \times m_2$ matrix M_2 if and only if there exists a subset $S \subseteq \text{cols}(M_2)$ of size t, such that the m_1 columns of M_1 are in the span of S. We say that a $\lambda n \times m_1$ matrix M_1 is *t-column block-constructible* from an $n \times m_2$ matrix M_2 if and only if for every $i \in [\varsigma]$, the submatrix $M_1[(i-1)n+1, \dots, in; *]$ is t-column constructible from M_2. ▶

Theorem 18: A systematic $(n^I, k^I; n^F, k^F = \varsigma k^I)$ convertible code with $k^I \times r^I$ initial parity generator matrix $\mathbf{P}^I$ and $k^F \times r^F$ final parity generator matrix $\mathbf{P}^F$ is MDS and access-optimal, if the following two conditions hold: (1) if $r^I \geq r^F$ then $\mathbf{P}^F$ is r^F-column block-constructible from $\mathbf{P}^I$, and (2) $\mathbf{P}^I, \mathbf{P}^F$ are superregular.

Proof: Follows from Proposition 2 and the fact that $\mathbf{P}^F$ must be generated by accessing just r^F symbols from each initial codeword (Lemma 12). ■

Thus, we can reduce the problem of proving the optimality of a systematic MDS convertible code in the merge regime to that of showing that matrices $\mathbf{P}^I$ and $\mathbf{P}^F$ satisfy the two properties mentioned in Theorem 18.

We first show that the construction specified in Section V-A satisfies condition (1) of Theorem 18.

Lemma 19: Let $\mathbf{P}^I, \mathbf{P}^F$ be as defined in Section V-A. Then $\mathbf{P}^F$ is r^F-column block-constructible from $\mathbf{P}^I$.

Proof: Consider the first r^F columns of $\mathbf{P}^I$, which we denote as $\mathbf{P}^I_{r^F} = \mathbf{P}^I[*; 1, \dots, r^F]$. Notice that $\mathbf{P}^F$ can be written as the following block matrix:

$$\mathbf{P}^F = \begin{bmatrix} \mathbf{P}^I_{r^F} \\ \mathbf{P}^I_{r^F}\,\text{diag}(1, \theta^{k^I}, \theta^{2k^I}, \dots, \theta^{(r^F-1)k^I}) \\ \mathbf{P}^I_{r^F}\,\text{diag}(1, \theta^{2k^I}, \theta^{2\cdot 2k^I}, \dots, \theta^{(r^F-1)2k^I}) \\ \vdots \\ \mathbf{P}^I_{r^F}\,\text{diag}(1, \theta^{(\varsigma-1)k^I}, \dots, \theta^{(r^F-1)(\varsigma-1)k^I}) \end{bmatrix},$$

where $\text{diag}(a_1, a_2, \dots, a_n)$ is the $n \times n$ diagonal matrix with $(a_1, \dots, a_n)$ as the diagonal elements. From this representation, it is clear that $\mathbf{P}^F$ can be constructed from the first r^F columns of $\mathbf{P}^I$. ■

It only remains to show that the construction in Section V-A satisfies condition (2) of Theorem 18, that is, that $\mathbf{P}^I$ and $\mathbf{P}^F$ are superregular.

Lemma 20: Let $\mathbf{P}^I, \mathbf{P}^F$ be as defined in Section V-A. Then $\mathbf{P}^I$ and $\mathbf{P}^F$ are superregular, for sufficiently large field size.

Proof: Let $\mathbf{R}$ be a $t \times t$ submatrix of $\mathbf{P}^I$ or $\mathbf{P}^F$, determined by the row indices $i_1 < i_2 < \cdots < i_t$ and the column indices $j_1 < j_2 < \cdots < j_t$, and denote entry (i, j) of $\mathbf{R}$ as $\mathbf{R}[i, j]$.

The determinant of $\mathbf{R}$ is defined by the Leibniz formula:

$$\begin{aligned}\det(\mathbf{R}) &= \sum_{\sigma\in\mathrm{Perm}(t)} \mathrm{sgn}(\sigma)\prod_{l=1}^{t}\mathbf{R}[l,\sigma(l)] \qquad (9)\\ &= \sum_{\sigma\in\mathrm{Perm}(t)} \mathrm{sgn}(\sigma)\theta^{E_\sigma}\\ \text{where}\quad E_\sigma &= \sum_{l=1}^{t}(i_l-1)(j_{\sigma(l)}-1),\end{aligned}$$

$\mathrm{Perm}(t)$ is the set of all permutations on t elements, and $\mathrm{sgn}(\sigma) \in \{-1,1\}$ is the sign of permutation σ. Clearly, $\det(\mathbf{R})$ defines a univariate polynomial $f_{\mathbf{R}} \in \mathbb{F}_p[\theta]$. We will now show that $\deg(f_{\mathbf{R}}) = \sum_{l=1}^{t}(i_l-1)(j_l-1)$ by showing that there is a unique permutation $\sigma^* \in \mathrm{Perm}(t)$ for which E_{σ^*} achieves this value, and that this is the maximum over all permutations in $\mathrm{Perm}(t)$. This means that $f_{\mathbf{R}}$ has a leading term of degree E_{σ^*}.

To prove this statement, we show that any permutation $\sigma \in \mathrm{Perm}(t)\backslash\{\sigma^*\}$ can be modified into a permutation σ' such that $E_{\sigma'} > E_\sigma$. Specifically, we show that $\sigma^* = \sigma_{\mathrm{id}}$, the identity permutation. Consider $\sigma \in \mathrm{Perm}(t)\backslash\{\sigma_{\mathrm{id}}\}$: let a be the smallest index such that $\sigma(a) \neq a$, let $b = \sigma^{-1}(a)$, and let $c = \sigma(a)$. Let σ' be such that $\sigma'(a) = a$, $\sigma'(b) = c$, and $\sigma'(d) = \sigma(d)$ for $d \in [t]\backslash\{a,b\}$. In other words, σ' is the result of "swapping" the images of a and b in σ. Notice that $a < b$ and $a < c$. Then, we have that:

$$\begin{aligned}E_{\sigma'} - E_\sigma &= (i_a-1)(j_a-1) + (i_b-1)(j_c-1)\\ &\quad - (i_a-1)(j_c-1) - (i_b-1)(j_a-1)\\ &= (i_b-i_a)(j_c-j_a) > 0\end{aligned}$$

The last inequality comes from the fact that $a < b$ implies $i_a < i_b$ and $a < c$ implies $j_a < j_c$. Therefore, $\deg(f_{\mathbf{R}}) = \max_{\sigma\in\mathrm{Perm}(t)} E_\sigma = E_{\sigma_{\mathrm{id}}}$.

Let $E^*(\varsigma, k^I, r^I, r^F)$ be the maximum degree of $f_{\mathbf{R}}$ over all submatrices $\mathbf{R}$ of $\mathbf{P}^I$ or $\mathbf{P}^F$. Then, $E^*(\varsigma, k^I, r^I, r^F)$ corresponds to the diagonal with the largest elements in $\mathbf{P}^I$ or $\mathbf{P}^F$. In $\mathbf{P}^F$ this is the diagonal of the square submatrix formed by the bottom r^F rows. In $\mathbf{P}^I$ it can be either the diagonal of the square submatrix formed by the bottom r^I rows, or by the right k^I columns. Thus, we have that:

$$E^*(\varsigma, k^I, r^I, r^F) = \max\{E_1, E_2, E_3\}$$

$$\begin{aligned}\text{where } E_1 &= \sum_{i=0}^{r^F-1} i(\varsigma k^I - r^F + i)\\ &= r^F(r^F-1)(3\varsigma k^I - r^F - 1)/6,\\ E_2 &= \sum_{i=0}^{r^I-1} i(k^I - r^I + i)\\ &= r^I(r^I-1)(3k^I - r^I - 1)/6,\\ E_3 &= \sum_{i=0}^{k^I-1} i(r^I - k^I + i)\\ &= k^I(k^I-1)(3r^I - k^I - 1)/6.\end{aligned}$$

Recall that we defined the field size as $q = p^D$ for any prime p. We set $D = (E^*(\varsigma, k^I, r^I, r^F) + 1)$. Then, if $\det(\mathbf{R}) = 0$ for some submatrix $\mathbf{R}$, θ is a root of $f_{\mathbf{R}}$, which is a contradiction since θ is a primitive element and the minimal polynomial of θ over $\mathbb{F}_p$ has degree $D > \deg(f_{\mathbf{R}})$ [92]. ■

Combining the above results leads to the following key result on the achievability of the lower bounds on access cost derived in Section IV.

Theorem 21: The explicit construction provided in Section V-A yields access-optimal linear MDS convertible codes for all parameter values in the merge regime.

Proof: Follows from Theorem 18, Lemma 19, and Lemma 20. ■

The construction presented in this section is practical only for small values of the parameters since the required field size grows exponentially with the lengths of the initial and final codes. In Section VI we present practical low-field-size constructions.

VI. Low Field-Size Convertible Codes Based on Superregular Hankel Arrays

In this section we present alternative constructions for $(n^I, k^I; n^F, k^F = \varsigma k^I)$ convertible code that require a significantly lower (polynomial) field size than the construction presented in Section V. We start by explaining the key ideas behind these constructions and present two examples that represent two extremes of a tradeoff between field size and coverage of parameter values. In Section VI-A, we describe the general construction, which includes codes at the two extremes of the tradeoff and a sequence of constructions in between. In Section VI-B, we show that the proposed code construction can support access-optimal conversion even when parameters of the final code are a priori unknown.

The key idea behind our constructions is to take the matrices $\mathbf{P}^I$ and $\mathbf{P}^F$ as cleverly-chosen submatrices from a specially constructed triangular array of the following form:

$$T_m : \quad \begin{array}{cccccc} b_1 & b_2 & b_3 & \cdots & b_{m-1} & b_m\\ b_2 & b_3 & \cdots & \cdots & b_m & \\ b_3 & \vdots & \iddots & \iddots & & \\ \vdots & \vdots & \iddots & & & \\ b_{m-1} & b_m & & & & \\ b_m & & & & & \end{array} \qquad (10)$$

with the property that every submatrix of T_m is superregular (the submatrix must lie completely within the triangular array). Here, (1) $(b_1, \ldots, b_m)$ are (not necessarily distinct) elements from $\mathbb{F}_q$, and (2) m is at most the field size q. The array T_m has Hankel form, that is, $T_m[i,j] = T_m[i-1, j+1]$, for all $i \in [2,m]$, $j \in [m-1]$. We denote T_m a *superregular Hankel array*. Such an array can be constructed by employing the algorithm proposed in [94] (where the algorithm was employed to generate generalized Cauchy matrices to construct generalized Reed-Solomon codes). This algorithm is described in Appendix for reference, although it is not necessary for understanding the constructions in this section.

We construct the initial and final codes by taking submatrices $\mathbf{P}^I$ and $\mathbf{P}^F$ from superregular Hankel arrays in a special

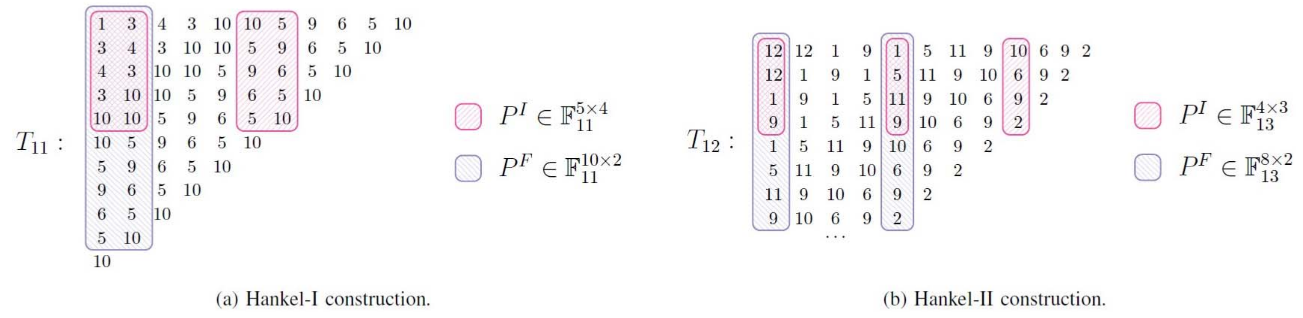


Fig. 5. Examples of constructions based on Hankel arrays: (a) Hankel-I construction parity generator matrices for systematic $(9, 5; 12, 10)$ convertible code. Notice how matrix $\mathbf{P}^F$ corresponds to the vertical concatenation of the first two columns and the last two columns of matrix $\mathbf{P}^I$. (b) Hankel-II construction parity generator matrices for systematic $(7, 4; 10, 8)$ convertible code. Notice how matrix $\mathbf{P}^F$ corresponds to the vertical concatenation of the first and second column of $\mathbf{P}^I$, and the second and third column of $\mathbf{P}^I$.

manner. This guarantees that $\mathbf{P}^I$ and $\mathbf{P}^F$ are superregular. In addition, we exploit the Hankel form of the array by carefully choosing the submatrices that form $\mathbf{P}^I$ and $\mathbf{P}^F$ to ensure that $\mathbf{P}^F$ is r^F-column block-constructible from $\mathbf{P}^I$. Given the way we construct these matrices and the properties of T_m, all the initial and final codes presented in this section turn out to be inside a well-studied class of codes known as (punctured) *generalized doubly-extended Reed-Solomon* codes [94].

The above idea yields a sequence of constructions with a tradeoff between the field size and the maximum value of r^F supported. We first present two examples that correspond to the extreme ends of this tradeoff, which we call *Hankel-I* and *Hankel-II*. Construction Hankel-I, shown in Example 22, can be applied whenever $r^F \le \lfloor r^I/\varsigma \rfloor$, and requires a field size of $q \ge (\max\{n^I, n^F\} - 1)$. Construction Hankel-II, shown in Example 23, can be applied whenever $r^F \le (r^I - \varsigma + 1)$, and requires a field size of $q \ge k^I r^I$.

Throughout this section we will assume that $\varsigma \le r^I \le k^I$. The ideas presented here are still applicable when $r^I > k^I$, but the constructions and analysis change in minor ways.

Example 22 (Hankel-I): Consider the parameters $(9, 5; 12, 10)$ and the field $\mathbb{F}_{11}$ (any finite field of size at least 11 suffices, but we choose a prime field for ease of explanation). Notice that these parameters satisfy:

$$r^F = 2 \le \left\lfloor \frac{r^I}{\varsigma} \right\rfloor = 2, \text{ and}$$
$$q = 11 \ge \max\{n^I, n^F\} - 1 = 11.$$

First, construct a superregular Hankel array of size $n^F - 1 = 11$, T_{11}, employing the algorithm in [94]. Then, divide the $r^I = 4$ initial parities into $\varsigma = 2$ groups: encoding vectors of parities in the same group will correspond to contiguous columns of T_{11}. The submatrix $\mathbf{P}^I \in \mathbb{F}_{11}^{5\times 4}$ is formed from the top $k^I = 5$ rows and columns $1, 2, k^I + 1 = 6$ and $k^I + 2 = 7$ of T_{11}, as shown in Figure 5a. The submatrix $\mathbf{P}^F \in \mathbb{F}_{11}^{10\times 2}$ is formed from the top $k^I = 10$ rows and columns $1, 2$ of T_{11}, as shown in Figure 5a. Checking that these matrices are superregular follows from the superregularity of T_{11}. It is straightforward to check that both these matrices are superregular, which follows from the superregularity of T_{11}. Furthermore, notice that the chosen parity matrices have the following structure:

$$\mathbf{P}^I = \begin{bmatrix} \top & \top & \top & \top \\ \mathbf{p_1} & \mathbf{p_2} & \mathbf{p_3} & \mathbf{p_4} \\ \bot & \bot & \bot & \bot \end{bmatrix}, \qquad \mathbf{P}^F = \begin{bmatrix} \top & \top \\ \mathbf{p_1} & \mathbf{p_2} \\ \bot & \bot \\ \top & \top \\ \mathbf{p_3} & \mathbf{p_4} \\ \bot & \bot \end{bmatrix}.$$

From this structure, it is clear that $\mathbf{P}^F$ is 2-column block-constructible from $\mathbf{P}^I$. Therefore, $\mathbf{P}^I$ and $\mathbf{P}^F$ satisfy the sufficient conditions of Theorem 18, and define an access-optimal convertible code. ▶

Example 23 (Hankel-II): Consider parameters $(7, 4; 10, 8)$ and field $\mathbb{F}_{13}$ (any finite field of size at least 12 suffices, but we choose a prime field for ease of explanation). Notice that these parameters satisfy:

$$r^F = 2 \le r^I - \varsigma + 1 = 2 \quad \text{and} \quad q = 13 \ge k^I r^I = 12$$

First, construct a superregular Hankel array of size $k^I r^I = 12$, T_{12}, by choosing $q = 13$ as the field size, and employing the algorithm in [94]. The submatrix $\mathbf{P}^I \in \mathbb{F}_{13}^{4\times 3}$ is formed by the top $k^I = 4$ rows and columns $\{1, (k^I + 1) = 5, (2k^I + 1) = 9\}$ of T_{12}, as shown in Figure 5b. The submatrix $\mathbf{P}^F \in \mathbb{F}_{13}^{8\times 2}$ is formed by the top $k^F = 8$ rows and columns $\{1, (k^I + 1) = 5\}$ of T_{12}, as shown in Figure 5b. It is easy to check that $\mathbf{P}^I$ and $\mathbf{P}^F$ are superregular, which follows from the superregularity of T_{12}. Furthermore, notice that the chosen parity matrices have the following structure:

$$\mathbf{P}^I = \begin{bmatrix} \top & \top & \top \\ \mathbf{p_1} & \mathbf{p_2} & \mathbf{p_3} \\ \bot & \bot & \bot \end{bmatrix}, \qquad \mathbf{P}^F = \begin{bmatrix} \top & \top \\ \mathbf{p_1} & \mathbf{p_2} \\ \bot & \bot \\ \top & \top \\ \mathbf{p_2} & \mathbf{p_3} \\ \bot & \bot \end{bmatrix}.$$

It is easy to see that $\mathbf{P}^F$ is 2-column block-constructible from $\mathbf{P}^I$. Therefore, $\mathbf{P}^I$ and $\mathbf{P}^F$ satisfy the sufficient conditions of Theorem 18, and define an access-optimal convertible code. ▶

A. General Hankel-Array-Based Construction of convertible codes

In this subsection, we present a sequence of Hankel-array-based constructions of access-optimal MDS convertible codes. This sequence of constructions presents a tradeoff between field size and the range of r^F supported. To index the sequence

$$\mathbf{P}^I = \left[\begin{array}{ccc|c|ccc|c|ccc} b_1 & \cdots & b_t & \cdots & b_{(i-1)k^I+1} & \cdots & b_{(i-1)k^I+t} & \cdots & b_{(s-1)k^I+1} & \cdots & b_{(s-1)k^I+t} \\ b_2 & \cdots & b_{t+1} & \cdots & b_{(i-1)k^I+2} & \cdots & b_{(i-1)k^I+t+1} & \cdots & b_{(s-1)k^I+2} & \cdots & b_{(s-1)k^I+t+1} \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ b_{k^I} & \cdots & b_{k^I+t-1} & \cdots & b_{ik^I} & \cdots & b_{ik^I+t-1} & \cdots & b_{sk^I} & \cdots & b_{sk^I+t-1} \end{array}\right]$$

$$\mathbf{P}^F = \left[\begin{array}{ccc|c|ccc|c|ccc} b_1 & \cdots & b_t & \cdots & b_{(i-1)k^I+1} & \cdots & b_{(i-1)k^I+t} & \cdots & b_{(s-\varsigma)k^I+1} & \cdots & b_{(s-\varsigma)k^I+t} \\ b_2 & \cdots & b_{t+1} & \cdots & b_{(i-1)k^I+2} & \cdots & b_{(i-1)k^I+t+1} & \cdots & b_{(s-\varsigma)k^I+2} & \cdots & b_{(s-\varsigma)k^I+t+1} \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ b_{\varsigma k^I} & \cdots & b_{\varsigma k^I+t-1} & \cdots & b_{(i+\varsigma-1)k^I} & \cdots & b_{(i+\varsigma-1)k^I+t-1} & \cdots & b_{sk^I} & \cdots & b_{sk^I+t-1} \end{array}\right].$$

Fig. 6. Generator matrix for initial and final parities in Hankel$_s$ construction. The vertical bars separate groups of columns. In matrix $\mathbf{P}^I$, the index i ranges from 1 to s. In matrix $\mathbf{P}^F$, the index i ranges from 1 to $(s - \varsigma + 1)$.

we use $s \in \{\varsigma, \varsigma+1, \ldots, r^I\}$ which corresponds to the number of groups into which the initial parity encoding vectors are divided. Given parameters $\{k^I, r^I, \varsigma\}$ and a field $\mathbb{F}_q$, construction Hankel$_s$ ($s \in \{\varsigma, \varsigma+1, \ldots, r^I\}$) supports:

$$r^F \leq (s-\varsigma+1)\left\lfloor \frac{r^I}{s} \right\rfloor + \max\{(r^I \bmod s) - \varsigma + 1, 0\},$$
$$\text{requiring} \quad q \geq \max\{sk^I + \left\lfloor \frac{r^I}{s} \right\rfloor - 1, n^I - 1\}.$$

Therefore, Hankel-I, from Example 22 corresponds to Hankel$_\varsigma$ and Hankel-II from Example 23 corresponds to Hankel$_{r^I}$.

1) Construction of Hankel$_s$: Assume, for the sake of simplicity, that $k^I \geq r^I$, $s \mid r^I$ and let $t = (r^I/s)$. Now we describe how to construct $\mathbf{P}^I$ and $\mathbf{P}^F$ over a field $\mathbb{F}_q$ whenever:

$$r^F \leq (s-\varsigma+1)t \quad \text{and} \quad q \geq sk^I + t - 1.$$

Without loss of generality, we consider $r^F = (s-\varsigma+1)t$ (lesser values of r^F can be obtained by puncturing the final code, i.e., eliminating some of the final parities). Let T_m be as in Equation 10, with $m = (sk^I + t - 1)$. Divide the r^I initial parity encoding vectors into s disjoint sets $(S_1, S_2, \ldots, S_s)$ of size t each. We associate each set S_i ($i \in [s]$) with a set of column indices $\mathrm{col}(S_i) = \{(i-1)k^I + 1, (i-1)k^I + 2, \ldots, (i-1)k^I + t\}$ of T_m. Matrix $\mathbf{P}^I$ is the submatrix formed by the top k^I rows and the columns indexed by the set $(\mathrm{col}(S_1) \cup \cdots \cup \mathrm{col}(S_s))$ of T_m. Matrix $\mathbf{P}^F$ is the submatrix formed by the top ςk^I rows and the columns indexed by the set $(\mathrm{col}(S_1) \cup \cdots \cup \mathrm{col}(S_{s-\varsigma+1}))$ of T_m. The resulting matrices $\mathbf{P}^I$ and $\mathbf{P}^F$ are shown in Figure 6. In the case where $s \nmid r^I$, we form an additional set S_{s+1} with the remaining (r^I mod s) initial parity encoding vectors, and proceed as above.

Theorem 24: Given parameters k^I, r^I, ς, and a field $\mathbb{F}_q$ Hankel$_s$ ($s \in \{\varsigma, \ldots, r^I\}$) constructs an access-optimal $(n^I, k^I; n^F, k^F = \varsigma k^I)$ convertible code if:

$$r^F \leq (s-\varsigma+1)\left\lfloor \frac{r^I}{s} \right\rfloor + \max\{(r^I \bmod s) - \varsigma + 1, 0\}$$
$$\text{and} \quad q \geq \max\{sk^I + \left\lfloor \frac{r^I}{s} \right\rfloor - 1, n^I - 1\}.$$

Proof: Consider the construction Hankel$_s$ described in this section, for some $s \in \{\varsigma, \ldots, r^I\}$. The Hankel form of T_m and the manner in which $\mathbf{P}^I$ and $\mathbf{P}^F$ are constructed guarantees that the l-th column of $\mathbf{P}^F$ corresponds to the vertical concatenation of columns $\{l, l+t, \ldots, l+(\varsigma-1)t\}$ of $\mathbf{P}^I$. Thus, $\mathbf{P}^F$ is r^F-column block-constructible from $\mathbf{P}^I$. Furthermore, since $\mathbf{P}^I$ and $\mathbf{P}^F$ are submatrices of T_m, they are superregular. Thus $\mathbf{P}^I$ and $\mathbf{P}^F$ satisfy both of the properties laid out in Theorem 18 and hence the convertible code constructed by Hankel$_s$ is access-optimal. ■

2) Conversion Procedure: During conversion, the k^I data symbols from each of the ς initial codewords remain unchanged, and become the $k^F = \varsigma k^I$ data symbols from the final codeword. The r^F new (parity) blocks from the final codeword are constructed by accessing symbols from the initial codewords as detailed below. To construct the l-th new symbol (corresponding to the l-th column of $\mathbf{P}^F$, $l \in [r^F]$), read parity symbol $(l + (i-1)t)$ from each initial codeword $i \in [\varsigma]$, and then sum the ς symbols read. The encoding vector of the new symbol will be equal to the sum of the encoding vectors of the symbols read. This is done for every new encoding vector $l \in [r^F]$.

B. Handling a Priori Unknown Parameters

In practice, the final parameters (n^F, k^F) might be unknown at the time of code construction, as they might depend on the empirically observed failure rates. Thus, it is of interest to construct initial codes that are (n^F, k^F)-access-optimally convertible for all (n^F, k^F) in a given set. The general construction and the Hankel-array based constructions presented above indeed provide such a property.

Proposition 25: Every initial code from an $(n^I, k^I; n^F, k^F = \varsigma k^I)$ convertible code constructed using the constructions in this section and Section V is also $(n^{F'}, k^{F'})$-access-optimally convertible for any $k^{F'} = \varsigma' k^I$ and $n^{F'} = (r^{F'} + k^{F'})$ with $0 \leq r^{F'} \leq r^F$ and $2 \leq \varsigma' \leq \varsigma$.

Proof: The conversion procedure can be easily modified to take fewer initial codewords (i.e. by treating some of the initial codewords as all-zero codewords) or construct fewer parity symbols. Since the access cost associated with each initial codeword is $\min\{k^I, r^I\}$, and the access cost associated with every parity symbol is $\varsigma+1$, the resulting conversion procedure has optimal access cost. ■

Thus, to support access-optimal conversion for all parameters $(n^F = \varsigma k^I + r^F, k^F = \varsigma k^I)$ in a given finite set of values for ς and r^F, it suffices to construct an access-optimal convertible code using the largest parameter ς and r^F in the set. Then, by Proposition 25, the initial code will support access-optimal conversion for all parameter values in the given set.

VII. Conclusion and Future Work

In this paper, we introduce the code conversion problem, and the framework of *convertible codes* which lays the theoretical foundation for the rigorous study of the code conversion problem. The proposed framework enables the investigation of the fundamental limits on the resource consumption of the code conversion operation, as shown by our derivation of the tight lower bounds on the access cost of conversions. Furthermore, we present explicit constructions of access-optimal convertible codes for a wide range of parameter regimes. These results show that it is indeed possible to achieve code conversion using significantly fewer accesses than the default approach for a wide range of parameter regimes.

Convertible codes have a significant potential for real-world impact by enabling resource-efficient redundancy tuning which has been shown to provide considerable cost benefits in large-scale cluster storage systems [9]. By laying the theoretical foundations for studying the code conversion problem, this work has opened up a wide new space for design of codes for storage systems with a host of open problems, such as studying different measures of conversion costs, studying conversion in non-MDS codes, and considering the efficiency of conversion in conjunction with other properties such as repair efficiency. Exploring these dimensions would include deriving fundamental limits on the chosen conversion cost, proving achievability results via code constructions meeting the lower bounds, and constructing practical, low-field-size convertible codes optimizing the conversion costs.

Appendix
Algorithm for Constructing Superregular Hankel Triangular Arrays

In this appendix we describe the algorithm from [94] for constructing a superregular Hankel triangular array over any finite field. This is provided as reference for completeness and is not necessary for understanding the constructions described in this paper. We note that the algorithm outlined in [94] takes the field size q as input, and generates T_q as the output. It is easy to see that T_q thus generated can be truncated to generate the triangular array T_m for any $m \le q$.

Let $\mathbb{F}_q$ be a given base field, and let $m \le q$ be the size of the output triangular array T_m. The triangular array T_m has Hankel form, as shown in Equation 10. Therefore, it suffices to specify the entries $b_1, b_2, \dots, b_m$ in the first column of T_m. On input $m \le q$, the algorithm proceeds as follows:

1) Consider the extension field $\mathbb{F}_{q^2}$ and choose an element $\beta \in \mathbb{F}_{q^2}$ such that $\beta^i \notin \mathbb{F}_q$ for $i \in [q]$ and $\beta^{q+1} \in \mathbb{F}_q$. Let $p(x) = x^2 + \mu\, x + \eta$ be the minimal polynomial of β over $\mathbb{F}_{q^2}$.
2) Let $\sigma_{-1}, \sigma_0, \dots, \sigma_m \in \mathbb{F}_q$ be such that $\sigma_{-1} = -\eta^{-1}, \sigma_0 = 0$, and $\sigma_i = -\mu\, \sigma_{i-1} - \eta\, \sigma_{i-2}$, for $i \in [m]$.
3) Set $b_i = \sigma_i^{-1}$, for all $i \in [m]$.

The resulting triangular array is superregular, that is, every square submatrix taken from T_m is superregular. Please refer to [94] for a proof of this fact.

References

[1] F. Maturana and K. V. Rashmi, "Convertible codes: New class of codes for efficient conversion of coded data in distributed storage," in *Proc. 11th Innov. Theor. Comput. Sci. Conf. (ITCS)*, vol. 151, T. Vidick, Ed. Seattle, WA, USA: Schloss Dagstuhl-Leibniz-Zentrum für Informatik, Jan. 2020, p. 66.

[2] S. Ghemawat, H. Gobioff, and S. Leung, "The Google file system," *ACM SIGOPS Operating Syst. Rev.*, vol. 37, no. 5, pp. 29–43, 2003.

[3] D. Borthakur, R. Schmidt, R. Vadali, S. Chen, and P. Kling. *HDFS RAID—Facebook (Presentation)*. Accessed: Jan. 6, 2022. [Online]. Available: http://www.slideshare.net/ydn/hdfs-raid-facebook

[4] C. Huang *et al.*, "Erasure coding in Windows Azure storage," in *Proc. USENIX Annu. Tech. Conf. (ATC)*, 2012, pp. 15–26.

[5] Apache Software Foundation. *Apache Hadoop Documentation: HDFS Erasure Coding*. Accessed: Jul. 23, 2019. [Online]. Available: https://hadoop.apache.org/docs/r3.0.0/hadoop-project-dist/hadoop-hdfs/HDFSErasureCoding.html

[6] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A solution to the network challenges of data recovery in erasure-coded distributed storage systems: A study on the Facebook warehouse cluster," in *Proc. 5th USENIX Workshop Hot Topics Storage File Syst. (HotStorage)*, Jun. 2013.

[7] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A 'hitchhiker's' guide to fast and efficient data reconstruction in erasure-coded data centers," in *Proc. ACM Conf. SIGCOMM*, 2014, pp. 331–342.

[8] M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "Xoring elephants: Novel erasure codes for big data," *Proc. VLDB Endowment*, vol. 6, no. 5, pp. 325–336, 2013.

[9] S. Kadekodi, K. V. Rashmi, and G. R. Ganger, "Cluster storage systems gotta have HeART: Improving storage efficiency by exploiting disk-reliability heterogeneity," in *Proc. 17th USENIX Conf. File Storage Technol. (USENIX FAST)*, 2019, pp. 345–358.

[10] S. Kadekodi, F. Maturana, S. J. Subramanya, J. Yang, K. V. Rashmi, and G. R. Ganger, "PACEMAKER: Avoiding HeART attacks in storage clusters with disk-adaptive redundancy," in *Proc. 14th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, Nov. 2020, pp. 369–385. [Online]. Available: https://www.usenix.org/conference/osdi20/presentation/kadekodi

[11] J. Huang, X. Liang, X. Qin, P. Xie, and C. Xie, "Scale-RS: An efficient scaling scheme for RS-coded storage clusters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 6, pp. 1704–1717, Jun. 2015.

[12] B. K. Rai, V. Dhoorjati, L. Saini, and A. K. Jha, "On adaptive distributed storage systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2015, pp. 1482–1486.

[13] M. Sonowal and B. K. Rai, "On adaptive distributed storage systems based on functional MSR code," in *Proc. Int. Conf. Wireless Commun., Signal Process. Netw. (WiSPNET)*, Mar. 2017, pp. 338–343.

[14] Y. Hu, X. Zhang, P. P. C. Lee, and P. Zhou, "Generalized optimal storage scaling via network coding," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Vail, CO, USA, Jun. 2018, pp. 956–960.

[15] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, no. 2, pp. 300–304, Jun. 1960.

[16] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (RAID)," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1988, pp. 109–116.

[17] J. Plank, "T1: Erasure codes for storage applications," in *Proc. 4th USENIX Conf. File Storage Technol.*, Jan. 2005, pp. 1–74.

[18] M. Blaum, J. Brady, J. Bruck, and J. Menon, "EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures," *IEEE Trans. Comput.*, vol. 44, no. 2, pp. 192–202, Feb. 1995.

[19] L. Xu and J. Bruck, "X-code: MDS array codes with optimal encoding," *IEEE Trans. Inf. Theory*, vol. 45, no. 1, pp. 272–276, Jan. 1999.

[20] C. Huang and L. Xu, "STAR: An efficient coding scheme for correcting triple storage node failures," *IEEE Trans. Comput.*, vol. 57, no. 7, pp. 889–901, Jul. 2008.

[21] J. L. Hafner, "WEAVER codes: Highly fault tolerant erasure codes for storage systems," in *Proc. 4th Conf. USENIX Conf. File Storage Technol.*, Berkeley, CA, USA, vol. 4, 2005, p. 16.

[22] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.

[23] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction," *IEEE Trans. Inf. Theory*, vol. 57, no. 8, pp. 5227–5239, Aug. 2011.

[24] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Interference alignment in regenerating codes for distributed storage: Necessity and code constructions," *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2134–2158, Apr. 2012.

[25] C. Suh and K. Ramchandran, "Exact-repair MDS code construction using interference alignment," *IEEE Trans. Inf. Theory*, vol. 57, no. 3, pp. 1425–1442, Mar. 2011.

[26] Z. Wang, I. Tamo, and J. Bruck, "On codes for optimal rebuilding access," in *Proc. 49th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2011, pp. 1374–1381.

[27] V. R. Cadambe, C. Huang, J. Li, and S. Mehrotra, "Polynomial length MDS codes with optimal repair in distributed storage," in *Proc. Conf. Rec. 45th Asilomar Conf. Signals, Syst. Comput. (ASILOMAR)*, Nov. 2011, pp. 1850–1854.

[28] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Distributed storage codes with repair-by-transfer and nonachievability of interior points on the storage-bandwidth tradeoff," *IEEE Trans. Inf. Theory*, vol. 58, no. 3, pp. 1837–1852, Mar. 2012.

[29] Z. Wang, I. Tamo, and J. Bruck, "Long MDS codes for optimal repair bandwidth," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2012, pp. 1182–1186.

[30] I. Tamo, Z. Wang, and J. Bruck, "Zigzag codes: MDS array codes with optimal rebuilding," *IEEE Trans. Inf. Theory*, vol. 59, no. 3, pp. 1597–1616, Mar. 2013.

[31] V. R. Cadambe, S. A. Jafar, H. Maleki, K. Ramchandran, and C. Suh, "Asymptotic interference alignment for optimal repair of MDS codes in distributed storage," *IEEE Trans. Inf. Theory*, vol. 59, no. 5, pp. 2974–2987, May 2013.

[32] D. Papailiopoulos, A. G. Dimakis, and V. Cadambe, "Repair optimal erasure codes through Hadamard designs," *IEEE Trans. Inf. Theory*, vol. 59, no. 5, pp. 3021–3037, May 2013.

[33] B. Sasidharan, G. K. Agarwal, and P. V. Kumar, "A high-rate MSR code with polynomial sub-packetization level," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2015, pp. 2051–2055.

[34] M. Ye and A. Barg, "Explicit constructions of optimal-access MDS codes with nearly optimal sub-packetization," *IEEE Trans. Inf. Theory*, vol. 63, no. 10, pp. 6307–6317, Oct. 2017.

[35] M. Ye and A. Barg, "Explicit constructions of high-rate MDS array codes with optimal repair bandwidth," *IEEE Trans. Inf. Theory*, vol. 63, no. 4, pp. 2001–2014, Apr. 2017.

[36] A. S. Rawat, O. O. Koyluoglu, and S. Vishwanath, "Progress on high-rate MSR codes: Enabling arbitrary number of helper nodes," in *Proc. Inf. Theory Appl. Workshop (ITA)*, Jan. 2016, pp. 1–6.

[37] B. Sasidharan, M. Vajha, and P. V. Kumar, "An explicit, coupled-layer construction of a high-rate MSR code with low sub-packetization level, small field size and $d < (n-1)$," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 2048–2052.

[38] S. Goparaju, A. Fazeli, and A. Vardy, "Minimum storage regenerating codes for all parameters," *IEEE Trans. Inf. Theory*, vol. 63, no. 10, pp. 6318–6328, Oct. 2017.

[39] A. Chowdhury and A. Vardy, "New constructions of MDS codes with asymptotically optimal repair," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 1944–1948.

[40] K. Mahdaviani, A. Khisti, and S. Mohajer, "Bandwidth adaptive & error resilient MBR exact repair regenerating codes," *IEEE Trans. Inf. Theory*, vol. 65, no. 5, pp. 2736–2759, May 2019.

[41] K. Mahdaviani, S. Mohajer, and A. Khisti, "Product matrix MSR codes with bandwidth adaptive exact repair," *IEEE Trans. Inf. Theory*, vol. 64, no. 4, pp. 3121–3135, Apr. 2018.

[42] A. S. Rawat, I. Tamo, V. Guruswami, and K. Efremenko, "MDS code constructions with small sub-packetization and near-optimal repair bandwidth," *IEEE Trans. Inf. Theory*, vol. 64, no. 10, pp. 6506–6525, Oct. 2018.

[43] N. B. Sha, K. V. Rashmi, and P. V. Kumar, "A flexible class of regenerating codes for distributed storage," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2010, pp. 1943–1947.

[44] K. W. Shum, "Cooperative regenerating codes for distributed storage systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2011, pp. 1–5.

[45] V. Abdrashitov, N. Prakash, and M. Medard, "The storage vs repair bandwidth trade-off for multiple failures in clustered storage networks," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Nov. 2017, pp. 46–50.

[46] K. Shanmugam, D. S. Papailiopoulos, A. G. Dimakis, and G. Caire, "A repair framework for scalar MDS codes," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 998–1007, May 2014.

[47] V. Guruswami and M. Wootters, "Repairing Reed–Solomon codes," in *Proc. 48th Annu. ACM Symp. Theory Comput.*, Jun. 2016, pp. 216–226.

[48] M. Ye and A. Barg, "Explicit constructions of MDS array codes and RS codes with optimal repair bandwidth," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2016, pp. 1202–1206.

[49] H. Dau and O. Milenkovic, "Optimal repair schemes for some families of full-length Reed–Solomon codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 346–350.

[50] I. Tamo, M. Ye, and A. Barg, "Optimal repair of Reed–Solomon codes: Achieving the cut-set bound," in *Proc. IEEE 58th Annu. Symp. Found. Comput. Sci. (FOCS)*, Oct. 2017, pp. 216–227.

[51] J. Mardia, B. Bartan, and M. Wootters, "Repairing multiple failures for scalar MDS codes," *IEEE Trans. Inf. Theory*, vol. 65, no. 5, pp. 2661–2672, May 2019.

[52] H. Dau, I. Duursma, H. M. Kiah, and O. Milenkovic, "Repairing Reed–Solomon codes with multiple erasures," *IEEE Trans. Inf. Theory*, vol. 64, no. 10, pp. 6567–6582, Oct. 2018.

[53] I. Tamo, M. Ye, and A. Barg, "The repair problem for Reed–Solomon codes: Optimal repair of single and multiple erasures with almost optimal node size," *IEEE Trans. Inf. Theory*, vol. 65, no. 5, pp. 2673–2695, May 2019.

[54] S. Balaji and P. V. Kumar, "A tight lower bound on the sub-packetization level of optimal-access MSR and MDS codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 2381–2385.

[55] K. V. Rashmi, P. Nakkiran, J. Wang, N. B. Shah, and K. Ramchandran, "Having your cake and eating it too: Jointly optimal erasure codes for I/O, storage, and network-bandwidth," in *Proc. 13th USENIX Conf. File Storage Technol. (FAST)*, 2015, pp. 81–94.

[56] I. Tamo, Z. Wang, and J. Bruck, "Access versus bandwidth in codes for storage," *IEEE Trans. Inf. Theory*, vol. 60, no. 4, pp. 2028–2037, Apr. 2014.

[57] S. Goparaju, I. Tamo, and R. Calderbank, "An improved sub-packetization bound for minimum storage regenerating codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2770–2779, May 2014.

[58] O. Alrabiah and V. Guruswami, "An exponential lower bound on the sub-packetization of MSR codes," in *Proc. 51st Annu. ACM SIGACT Symp. Theory Comput.*, New York, NY, USA, Jun. 2019, pp. 979–985.

[59] K. Rashmi, N. B. Shah, and K. Ramchandran, "A piggybacking design framework for read-and download-efficient distributed storage codes," *IEEE Trans. Inf. Theory*, vol. 63, no. 9, pp. 5802–5820, Sep. 2017.

[60] V. Guruswami and A. S. Rawat, "MDS code constructions with small sub-packetization and near-optimal repair bandwidth," in *Proc. 28th Annu. ACM-SIAM Symp. Discrete Algorithms*, Jan. 2017.

[61] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the locality of codeword symbols," *IEEE Trans. Inf. Theory*, vol. 58, no. 11, pp. 6925–6934, Nov. 2011.

[62] A. S. Rawat, O. O. Koyluoglu, N. Silberstein, and S. Vishwanath, "Optimal locally repairable and secure codes for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 60, no. 1, pp. 212–236, Nov. 2013.

[63] J. Katz and L. Trevisan, "On the efficiency of local decoding procedures for error-correcting codes," in *Proc. 32nd Annu. ACM Symp. Theory Comput.*, F. F. Yao and E. M. Luks, Eds. Portland, OR, USA: ACM Press, May 2000, pp. 80–86.

[64] M. Blaum, J. L. Hafner, and S. Hetzler, "Partial-MDS codes and their application to RAID type of architectures," *IEEE Trans. Inf. Theory*, vol. 59, no. 7, pp. 4510–4519, Jul. 2013.

[65] P. Gopalan, C. Huang, B. Jenkins, and S. Yekhanin, "Explicit maximally recoverable codes with locality," *IEEE Trans. Inf. Theory*, vol. 60, no. 9, pp. 5245–5256, Sep. 2014.

[66] D. S. Papailiopoulos and A. G. Dimakis, "Locally repairable codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 10, pp. 5843–5855, Oct. 2014.

[67] I. Tamo and A. Barg, "A family of optimal locally recoverable codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 8, pp. 4661–4676, May 2014.

[68] G. M. Kamath, N. Prakash, V. Lalitha, and P. V. Kumar, "Codes with local regeneration and erasure correction," *IEEE Trans. Inf. Theory*, vol. 60, no. 8, pp. 4637–4660, Aug. 2014.

[69] V. R. Cadambe and A. Mazumdar, "Bounds on the size of locally recoverable codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 11, pp. 5787–5794, Nov. 2015.

[70] I. Tamo, D. S. Papailiopoulos, and A. G. Dimakis, "Optimal locally repairable codes and connections to matroid theory," *IEEE Trans. Inf. Theory*, vol. 62, no. 12, pp. 6661–6671, Dec. 2016.

[71] I. Tamo, A. Barg, and A. Frolov, "Bounds on the parameters of locally recoverable codes," *IEEE Trans. Inf. Theory*, vol. 62, no. 6, pp. 3070–3083, Jun. 2016.

[72] A. Barg, K. Haymaker, E. W. Howe, G. L. Matthews, and A. Várilly-Alvarado, "Locally recoverable codes from algebraic curves and surfaces," in *Algebraic Geometry for Coding Theory and Cryptography*. Cham, Switzerland: Springer, 2017, pp. 95–127.

[73] S. L. Frank-Fischer, V. Guruswami, and M. Wootters, "Locality via partially lifted codes," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*. Wadern, Germany: Schloss Dagstuhl-Leibniz-Zentrum füer Informatik, 2017.

[74] A. Agarwal, A. Barg, S. Hu, A. Mazumdar, and I. Tamo, "Combinatorial alphabet-dependent bounds for locally recoverable codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 5, pp. 3481–3492, May 2018.

[75] A. Mazumdar, "Capacity of locally recoverable codes," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Nov. 2018, pp. 1–5.

[76] V. Guruswami, C. Xing, and C. Yuan, "How long can optimal locally repairable codes be?" in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*. Wadern, Germany: Schloss Dagstuhl-Leibniz-Zentrum füer Informatik, 2018.

[77] S. Gopi, V. Guruswami, and S. Yekhanin, "Maximally recoverable LRCs: A field size lower bound and constructions for few heavy parities," in *Proc. 13th Annu. ACM-SIAM Symp. Discrete Algorithms*. Philadelphia, PA, USA: SIAM, 2019, pp. 2154–2170.

[78] G. Zhang, W. Zheng, and J. Shu, "ALV: A new data redistribution approach to RAID-5 scaling," *IEEE Trans. Comput.*, vol. 59, no. 3, pp. 345–357, Mar. 2010.

[79] W. Zheng and G. Zhang, "FastScale: Accelerate RAID scaling by minimizing data migration," in *Proc. 9th USENIX Conf. File Storage Technol.*, G. R. Ganger and J. Wilkes, Eds. San Jose, CA, USA: USENIX Association, Feb. 2011, pp. 149–161. [Online]. Available: http://www.usenix.org/events/fast11/tech/techAbstracts.html

[80] C. Wu and X. He, "GSR: A global stripe-based redistribution approach to accelerate RAID-5 scaling," in *Proc. 41st Int. Conf. Parallel Process.*, Pittsburgh, PA, USA, Sep. 2012, pp. 460–469.

[81] G. Zhang, W. Zheng, and K. Li, "Rethinking RAID-5 data layout for better scalability," *IEEE Trans. Comput.*, vol. 63, no. 11, pp. 2816–2828, Nov. 2014.

[82] S. Wu, Y. Xu, Y. Li, and Z. Yang, "I/O-efficient scaling schemes for distributed storage systems with CRS codes," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 9, pp. 2639–2652, Sep. 2016.

[83] X. Zhang, Y. Hu, P. P. C. Lee, and P. Zhou, "Toward optimal storage scaling via network coding: From theory to practice," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Honolulu, HI, USA, Apr. 2018, pp. 1808–1816.

[84] X. Y. Zhang and Y. C. Hu, "Efficient storage scaling for MBR and MSR codes," *IEEE Access*, vol. 8, pp. 78992–79002, 2020.

[85] B. K. Rai, "On adaptive (functional MSR code based) distributed storage systems," in *Proc. Int. Symp. Netw. Coding (NetCod)*, Jun. 2015, pp. 46–50.

[86] S. Wu, Z. Shen, and P. P. C. Lee, "On the optimal repair-scaling trade-off in locally repairable codes," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Jul. 2020, pp. 2155–2164.

[87] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Enabling node repair in any erasure code for distributed storage," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2011, pp. 1235–1239.

[88] S. Mousavi, T. Zhou, and C. Tian, "Delayed parity generation in MDS storage codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Vail, CO, USA, Jun. 2018, pp. 1889–1893.

[89] M. Xia, M. Saxena, M. Blaum, and D. Pease, "A tale of two erasure codes in HDFS," in *Proc. 13th USENIX Conf. File Storage Technol. (FAST)*, J. Schindler and E. Zadok, Eds. Santa Clara, CA, USA: USENIX Association, Feb. 2015, pp. 213–226. [Online]. Available: https://www.usenix.org/conference/fast15/technical-sessions/presentation/xia

[90] X. Su, X. Zhong, X. Fan, and J. Li, "Local re-encoding for coded matrix multiplication," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Los Angeles, CA, USA, Jun. 2020, pp. 221–226.

[91] S. Wu, Z. Shen, and P. P. C. Lee, "Enabling I/O-efficient redundancy transitioning in erasure-coded KV stores via elastic Reed–Solomon codes," in *Proc. Int. Symp. Reliable Distrib. Syst. (SRDS)*, Shanghai, China, Sep. 2020, pp. 246–255.

[92] F. MacWilliams and N. Sloane, *The Theory Error-Correcting Codes*, 2nd ed. Amsterdam, The Netherlands: North Holland, 1978.

[93] H. Gluesing-Luerssen, J. Rosenthal, and R. Smarandache, "Strongly-MDS convolutional codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 2, pp. 584–598, Feb. 2006.

[94] R. M. Roth and G. Seroussi, "On generator matrices of MDS codes (Corresp.)," *IEEE Trans. Inf. Theory*, vol. 31, no. 6, pp. 826–830, Nov. 1985.

Francisco Maturana (Student Member, IEEE) received the B.S. and M.S. degrees in computer science from the Pontificia Universidad Católica de Chile, Santiago, Chile, in 2017. He is currently pursuing the Ph.D. degree with the Computer Science Department, Carnegie Mellon University, USA. His research interests lie at the intersection of theoretical computer science and computer systems.

K. V. Rashmi (Member, IEEE) received the Ph.D. degree from the UC Berkeley in 2016. She was a Post-Doctoral Scholar at UC Berkeley from 2016 to 2017. She is an Assistant Professor with the Computer Science Department, Carnegie Mellon University. Her research interests broadly lie in information/coding theory and computer/networked systems. During her Ph.D. studies, she was a recipient of the Facebook Fellowship from 2012 to 2013, the Microsoft Research Ph.D. Fellowship from 2013 to 2015, and the Google Anita Borg Memorial Scholarship from 2015 to 2016. She was also a recipient of the VMWare Systems Research Award 2021, the NSF CAREER Award 2020–2025, the Tata Institute of Fundamental Research Memorial Lecture Award 2020, the Facebook Distributed Systems Research Award 2019, the Google Faculty Research Award 2018, and the Facebook Communications and Networking Research Award 2017. Her Ph.D. thesis was awarded the UC Berkeley Eli Jury Dissertation Award 2016, and her work has received the USENIX NSDI 2021 Community (Best Paper) Award, and the IEEE Data Storage Best Paper and the Best Student Paper Awards for the years 2011/2012.