Convertible Codes: New Class of Codes for Efficient Conversion of Coded Data in Distributed Storage

Francisco Maturana
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
http://www.cs.cmu.edu/~fmaturan/
fmaturan@cs.cmu.edu

K. V. Rashmi
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
http://www.cs.cmu.edu/~rvinayak/
rvinayak@cs.cmu.edu

Abstract

Erasure codes are typically used in large-scale distributed storage systems to provide durability of data in the face of failures. In this setting, a set of k blocks to be stored is encoded using an $[n, k]$ code to generate n blocks that are then stored on different storage nodes. A recent work by Kadekodi et al. [32] shows that the failure rate of storage devices vary significantly over time, and that changing the rate of the code (via a change in the parameters n and k) in response to such variations provides significant reduction in storage space requirement. However, the resource overhead of realizing such a change in the code rate on already encoded data in traditional codes is prohibitively high.

Motivated by this application, in this work we first present a new framework to formalize the notion of *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, such as the maximum-distance-separable (MDS) property. We then introduce *convertible codes*, a new class of code pairs that allow for code conversions in a resource-efficient manner. For an important parameter regime (which we call the merge regime) along with the widely used linearity and MDS decodability constraint, we prove tight bounds on the number of nodes accessed during code conversion. In particular, our achievability result is an explicit construction of MDS convertible codes that are optimal for all parameter values in the merge regime albeit with a high field size. We then present explicit low-field-size constructions of optimal MDS convertible codes for a broad range of parameters in the merge regime. Our results thus show that it is indeed possible to achieve code conversions with significantly lesser resources as compared to the default approach of re-encoding.

2012 ACM Subject Classification Mathematics of computing → Coding theory; Theory of computation → Error-correcting codes

Keywords and phrases Coding theory, Reed-Solomon codes, Erasure codes, Code conversion, Distributed storage

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.66

Funding This work was funded in part by a Google Faculty Research Award.

Acknowledgements We thank Venkatesan Guruswami and Michael Rudow for their helpful suggestions on improving the writing for the paper.

1 Introduction

Erasure codes have become an essential tool for protecting against failures in distributed storage systems [18, 9, 30, 4]. 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 will contain several independent codewords distributed across different sets of storage nodes in the cluster.

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 [32]. For example, in [32], the authors show that an 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 [32]. We refer the reader to [32] for a more detailed discussion on the practical benefits and constraints of adapting the erasure-code parameters with the variations in failure rates in a storage system.

Changing n and k for stripes in a storage system would involve converting already encoded data from one code to another. Such conversions, however, can generate a large amount of load (as explained below) which adversely affects the operation of the cluster. Furthermore, in some cases these 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. Hence it is critical to minimize the resource consumption of code conversion operations. Clearly, it is always possible to re-encode the data in a codeword (or a stripe) according to a new code by accessing (or decoding if needed) all the (original) message symbols. However, such an approach, which we call the *default approach*, requires accessing a large number of nodes (for example, for MDS codes, the initial value of k number of nodes need to be accessed) reading out all the data, transferring over the network, and re-encoding, which consumes large amounts of resources.

To the best of our knowledge, the existing literature [44, 59, 29] on formally studying the problem of changing parameters n and k for already encoded data model the code conversion problem within the framework of *repair-efficient* codes. Repair-efficient codes (e.g., [16, 47, 71, 49, 21]), which are a class of codes that can reconstruct a small subset of codeword symbols more efficiently than reconstructing the entire (original) message, has been an active area of research in the recent past. While the existing approach of exploiting repair efficiency for conversion efficiency provides reduction in the network bandwidth consumed as compared to the default approach for several parameter regimes, it has several drawbacks. For example, under this approach conversion requires accessing every symbol in the codeword and redistributing ("subsymbols") around. A more detailed discussion on repair-efficient codes and existing works which exploit repair efficiency for conversion efficiency is provided in Section 6.

In this paper, we propose a fundamentally new framework to model the code conversion problem, which is independent of repair efficiency. Our approach is based on the observation 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

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

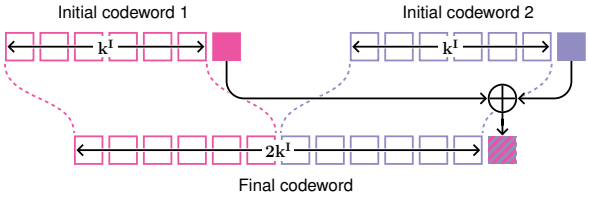


Figure 1 Example of code conversion: two codewords of a $[k^I + 1, k^I]$ single-parity-check code become one codeword of a $[2k^I + 1, 2k^I]$ single-parity-check code. The parity symbols are shown shaded. The data symbols from each codeword are preserved, and the parity symbol from the final codeword is the sum of the parities from each initial codeword.

codes satisfying the maximum-distance-separability (MDS) property. The code constructions that we present in this paper, under this new framework, require accessing significantly fewer symbols during conversion than existing works. Furthermore, even though the constructions presented in this paper optimize for reducing the number of symbols accessed for conversion, the bandwidth consumed for conversion is also lower in several parameter regimes of practical interest, compared to existing works (based on repair efficiency) that optimize for network bandwidth.

We now present a toy example to elucidate the concept of code conversion in our framework.

▶ **Example 1.** Consider conversion from an $[n^I = k^I + 1, k^I]$ code $\mathcal{C}^I$ to an $[n^F = k^F + 1, k^F]$ code $\mathcal{C}^F$. In our framework, this conversion is achieved by "merging" two codewords of the initial code for each codeword of the final code. Let us focus on the number of symbols accessed during conversion. The default approach requires accessing k^I symbols from each codeword belonging to $\mathcal{C}^I$ (initial codewords), and accessing at least one symbol for each codeword belonging to $\mathcal{C}^F$ (final codewords) to write out the result, totalling $2k^I + 1$ symbols accessed per final codeword. Alternatively, as depicted in Figure 1, one can choose $\mathcal{C}^I$ and $\mathcal{C}^F$ to be single-parity-check codes with the parity symbol holding the sum of the data symbols in each codeword (shown with a shaded box in the figure). To convert from $\mathcal{C}^I$ to $\mathcal{C}^F$, one sums the parity symbol from each initial codeword and stores the result as the parity symbol of the final codeword $\mathcal{C}^F$. This alternative approach requires accessing only three symbols for each final codeword, which is significantly more efficient.

Next, we give an overview of the main results and key ideas presented in this paper.

1.1 Overview of results

In this paper, we propose a *new framework to model the problem of code conversion*, that is, 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, such as maximum-distance-separable (MDS) property (Section 2). We then introduce a new class of code pairs, which we call *convertible codes*, that allow for resource-efficient conversions. We begin the study of this new class of code pairs, by focusing on an important regime where $k^F = \lambda k^I$ for any integer $\lambda \geq 2$ with arbitrary values of n^I and n^F, which we call the *merge regime*. Furthermore, we focus on the *access cost* of code conversion, which corresponds to *the total number of symbols accessed during conversion*. Keeping the number of symbols accessed small makes conversion less disruptive, allows the unaffected nodes to remain available for normal operations, and also reduces the amount of computation and communication needed.

Using our framework, we prove a tight lower bound on the access cost of conversions for linear MDS codes in the merge regime (Section 3). Let $r^I = n^I - k^I$ and $r^F = n^F - k^F$. This lower bound identifies two regions: (1) $r^F \leq r^I$, where significant savings in access cost can be achieved when $r^F < k^I$, and (2) the complement $r^F > r^I$, where linear MDS codes cannot achieve lower access cost than the default approach. Specifically, we prove:

▶ **Theorem 15.** *For all linear MDS $(n^I, k^I; n^F, k^F = \lambda k^I)$ convertible codes, the access cost of conversion is at least $r^F + \lambda \min\{k^I, r^F\}$. Furthermore, if $r^I < r^F$, the access cost of conversion is at least $r^F + \lambda k^I$.*

We prove this lower bound first on a class of linear MDS convertible codes that we call *stable* convertible codes. We then extend this lower bound to all linear MDS convertible codes by showing that non-stable convertible codes cannot achieve this lower bound, and thus all optimal linear MDS convertible codes in the merge regime are stable.

Then, we describe a construction of linear MDS Convertible Codes in the merge regime that meet the bound of Theorem 15, thereby providing convertible codes that are access-optimal (Section 4). Specifically:

▶ **Theorem 22.** *The explicit construction provided in Section 4.1 yields access-optimal linear MDS convertible codes for all parameter values in the merge regime.*

The construction is deterministic and yields codes over a finite extension field. To prove that those codes are access-optimal, we show that the resulting generator matrices have a particular structure, and that none of their minors are zero when the degree of the field extension is large enough. This implies that the field size required is very large (exponential in n^F for fixed constant final code rate and typical parameters). Field size is an important aspect of code design that has been widely studied in the context of other code families [12, 62, 2, 24]. To address this issue, we introduce a sequence of constructions of convertible codes that have significantly lower field size requirements (Section 5). This sequence of constructions, which we denote Hankel$_s$ ($s \in \{\lambda, \ldots, r^I\}$), presents a tradeoff between field size and the parameter range that they support. Specifically, we show:

▶ **Theorem 25.** *Given parameters k^I, r^I, λ, and a field $\mathbb{F}_q$, Hankel$_s$ ($s \in \{\lambda, \ldots, r^I\}$) constructs an access-optimal $(n^I, k^I; n^F, k^F = \lambda k^I)$ convertible code if:*

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

Of particular interest are the two extreme points of this tradeoff which we call Hankel-I and Hankel-II respectively: Hankel-I supports $r^F \leq \lfloor r^I/\lambda \rfloor$ requiring field size $q \geq \max\{n^I - 1, n^F - 1\}$; and Hankel-II supports a wider range $r^F \leq r^I - \lambda + 1$ requiring field size $q \geq k^I r^I$.

The key idea behind our construction is to construct the parity generator matrices of the initial and final codes as cleverly-chosen submatrices of a specially constructed upper-left triangular array. This array has two important properties: (1) every square submatrix of this array is non-singular, and (2) it has *Hankel form*, that is, each ascending diagonal from left to right is constant. By exploiting the repetitive structure of the array, our constructions yield convertible codes that achieve optimal access cost during conversion. Given the way in which these codes are constructed, they also have the additional property of being (punctured) generalized doubly-extended Reed-Solomon codes.

Our results thus show that there is a broad regime where code conversions can be achieved with significantly less access cost than the default approach, and another regime where achieving less access cost than the default approach is not possible. We provide a

general explicit construction of access-optimal convertible codes in the former regime, and low-field-size constructions for a wide parameter range within this regime. The framework of convertible codes presented in this work opens several new opportunities which pose interesting theoretical questions that have a potential for impact on real-world systems.

1.2 Background

In this subsection we introduce some notation and basic definitions 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 code $\mathcal{C}$ to a specific generator matrix $\mathbf{G}$, which will be clear from the 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 $[i]$ denote the set $\{1, 2, \ldots, i\}$. 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 [34].

2 A framework for studying code conversions

In this section, we formally define our new framework for studying *code conversions* and introduce *convertible codes*. While we use the notation of linear codes introduced in Section 1.2, 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 $M = \text{lcm}(k^I, k^F)$ 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 multiple codewords in the initial and final configurations. 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 problem (e.g., [16, 26, 47, 49]) and code locality (e.g., [21, 41, 27]), 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 $\mathcal{P}_I \subseteq 2^{[M]}$ is composed of M/k^I disjoint subsets of size k^I, and the final partition $\mathcal{P}_F \subseteq 2^{[M]}$ is composed of M/k^F disjoint subsets of size k^F. In the initial (respectively, final) configuration, the data indexed by each subset $S \in \mathcal{P}_I$ (respectively, $\mathcal{P}_F$) is encoded using the code $\mathcal{C}^I$ (respectively, $\mathcal{C}^F$). The codewords $\{\mathcal{C}^I(\mathbf{m}_S),\ S \in \mathcal{P}_I\}$ are referred to as *initial codewords*, and the codewords $\{\mathcal{C}^F(\mathbf{m}_S),\ S \in \mathcal{P}_F\}$ are referred to as *final codewords*, where $\mathbf{m}_S$ corresponds to the projection of $\mathbf{m}$ onto the coordinates in S. 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.

▶ **Definition 2** (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 $\{\mathcal{C}^I(\mathbf{m}_S) \mid S \in \mathcal{P}_I\}$ as input, and outputs the corresponding set of final codewords $\{\mathcal{C}^F(\mathbf{m}_S) \mid S \in \mathcal{P}_F\}$.*

▶ **Definition 3** (Convertible code). *A $(n^I, k^I; n^F, k^F)$ 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 $[M = \text{lcm}(k^I, k^F)]$ 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 $\{\mathcal{C}^I(\mathbf{m}_S) \mid S \in \mathcal{P}_I\}$ outputs $\{\mathcal{C}^F(\mathbf{m}_S) \mid S \in \mathcal{P}_F\}$ for all $\mathbf{m} \in \mathbb{F}_q^M$.*

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

▶ Remark 4. Note that the definition of convertible codes (Definition 3) assumes that $(n^I, k^I; n^F, k^F)$ are known at the time of code construction. This will be helpful in understanding the fundamental limits of the conversion process. In practice, this assumption might not hold. For example, n^F, k^F might depend on the node failure rates that are yet to be observed. *Interestingly, it is possible for a $(n^I, k^I; n^F, k^F)$ convertible code to facilitate conversion to multiple values of n^F, k^F simultaneously, as is the case for the code constructions presented in this paper.*

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 3 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 what follows, we will focus on a specific regime and a specific cost model.

3 Lower bounds on access cost of code conversion

The focus of this section is on deriving lower bounds on the access cost of code conversion in the merge regime (as defined below). We focus on a fundamental regime given by $k^F = \lambda k^I$, where integer $\lambda \geq 2$ is the number of initial codewords merged, with arbitrary values of n^I and n^F. We call this regime as *merge regime*. We additionally require that both the initial and final code are linear and MDS. Since linear MDS codes are widely used in storage systems and are well understood in the Coding Theory literature, they constitute a good starting point. In terms of cost of conversion, we focus on the *access cost* of code conversion, that is, the number of symbols that are affected by the conversion. Each symbol read from the initial codewords requires one symbol access and each symbol written to the final codeword requires one symbol access. Therefore, minimizing access cost amounts to *minimizing the sum of the number of symbols written to the final codeword and the number of symbols read from the initial codewords.*[2] Keeping this number small makes code conversion less disruptive

[2] 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.

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.

▶ Remark 5. Note that, when defining the access cost above, we implicitly assume that conversion is performed by downloading all the required data to a central location, where the new symbols are computed and then distributed to their final locations. This assumption does not affect the access cost, but it could affect other forms of cost, such as network bandwidth, which could be reduced by transferring data in a different way.

We now introduce some notation. We use the term *unchanged symbols* to refer to symbols from the initial codewords that remain *as is* in the final codeword, and we use the term *new symbols* to refer to symbols from the final codeword that are not present in the initial codewords (i.e. they are not unchanged). For example, in Figure 1, all the data symbols are unchanged symbols (unshaded boxes), and the parity symbol of the final codeword is a new symbol (striped box). We define the *read access set* of an $(n^I, k^I; n^F, k^F = \lambda k^I)$ convertible code as a set of tuples $\mathcal{D} \in [\lambda] \times [n^I]$, where $(i, j) \in \mathcal{D}$ corresponds to the j-th symbol of initial codeword i. The set $\mathcal{D}$ must be such that all new symbols are linear combinations of symbols indexed by the tuples in $\mathcal{D}$. Furthermore, we use $\mathcal{D}_i = \{j \mid (i, j) \in \mathcal{D}\}$, $\forall i \in [\lambda]$ to denote the symbols read from a particular initial codeword.

In Section 4, we show that the lower bounds on the access cost derived in this section are in fact achievable. Therefore, we refer to MDS convertible codes in the merge regime achieving these lower bounds as *access-optimal*.

▶ **Definition 6** (Access-optimal). *A linear MDS $(n^I, k^I; n^F, k^F = \lambda k^I)$ convertible code is said to be* access-optimal *if and only if it attains the minimum access cost over all linear MDS $(n^I, k^I; n^F, k^F = \lambda k^I)$ convertible codes.*

Now we present the access cost lower bounds of convertible codes in the merge regime.

3.1 Lower bounds on the access cost of code conversion

In this subsection, we present lower bounds on the access cost of linear MDS convertible codes in the merge regime. This is done in four steps:

1. We show that in the merge regime, all possible pairs of partitions $\mathbf{P}^I$ and $\mathbf{P}^F$ partitions are equivalent up to relabeling, and hence do not need to be specified.
2. An upper bound on the maximum number of unchanged symbols is proved. We call convertible codes that meet this bound as "*stable*".
3. Lower bounds on the access cost of linear MDS convertible codes are proved, under the added restriction that the convertible codes are stable.
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.

In the general regime, partitions need to be specified since they indicate how message symbols from the initial codewords are mapped into the final codewords. In the merge regime, however, the choice of the partitions does not matter.

▶ **Proposition 7.** *For every $(n^I, k^I; n^F, k^F = \lambda k^I)$ convertible code, all possible pairs of initial and final partitions $(\mathcal{P}_I, \mathcal{P}_F)$ are equivalent up to relabeling.*

Proof. It holds that $M = \lambda k^I$, and there is only one possible final partition $\mathcal{P}_F = \{[\lambda k^I]\}$. Thus, all data is mapped to the same final codeword, regardless of $\mathcal{P}_I$. ◀

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.

▶ **Proposition 8.** *In an MDS* $(n^I, k^I; n^F, k^F = \lambda k^I)$ *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. ◀

This implies that there are at most λk^I unchanged symbols and at least r^F new symbols in total.

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 convertible codes that minimize the number of new symbols, which we call *stable*.

▶ **Definition 9** (Stability)**.** *An MDS* $(n^I, k^I; n^F, k^F = \lambda k^I)$ *convertible code is* stable *if and only if it has exactly* λk^I *unchanged symbols, or in other words, exactly* r^F *new blocks.*

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. Minimizing the access cost of a stable convertible code reduces to minimizing the size of its read access set $\mathcal{D}$. The first lower bound on the size of $\mathcal{D}_i$ is given by the interaction between new symbols and the MDS property.

▶ **Lemma 10.** *For all linear stable MDS* $(n^I, k^I; n^F, k^F = \lambda k^I)$ *convertible codes, the read access set* $\mathcal{D}_i$ *from each initial codeword* $i \in [\lambda]$ *satisfies* $|\mathcal{D}_i| \geq \min\{k^I, r^F\}$.

Proof sketch. For the MDS property to hold in the final code, the encoding vectors of the new symbols must fulfill certain independence requirements. This requires reading at least as many symbol from each initial codeword as there are new symbols, up to k^I symbols. Please refer to Appendix A.2 for full proof. ◀

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 11.** *For all linear stable MDS* $(n^I, k^I; n^F, k^F = \lambda k^I)$ *convertible codes, if* $r^I < r^F$ *then the read access set* $\mathcal{D}_i$ *from each initial codeword* $i \in [\lambda]$ *satisfies* $|\mathcal{D}_i| \geq k^I$.

Proof sketch. By Lemma 10, one is forced to read at least $r^I + 1$ symbols. Hence there exist symbols that are both unchanged and are read during conversion. Since unchanged blocks are also part of the final codeword, the information read from these symbols is not useful in creating a new symbol that retains the MDS property of the final code, unless k^I symbols (that is, full data) are read. Please refer to Appendix A.3 for full proof. ◀

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 12.** *Let* $d^*(n^I, k^I; n^F, k^F)$ *denote the minimum integer* d *such that there exists a linear stable MDS* $(n^I, k^I; n^F, k^F = \lambda k^I)$ *convertible code with read access set* $\mathcal{D}$ *of size* $|\mathcal{D}| = d$*. For all valid parameters,* $d^*(n^I, k^I; n^F, k^F) \geq \lambda \min\{k^I, r^F\}$*. Furthermore, if* $r^I < r^F$*, then* $d^*(n^I, k^I; n^F, k^F) \geq \lambda k^I$.

Proof. Follows directly from Lemma 10 and Lemma 11. ◀

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 13.** *The lower bounds on the size of the read access set from Theorem 12 hold for* ***all*** *linear MDS* $(n^I, k^I; n^F, k^F = \lambda k^I)$ *convertible codes.*

Proof. In Appendix A.4. ◀

The above result, along with the fact that the lower bound in Theorem 12 is achievable (as will be shown in Section 4), implies that all access-optimal linear MDS convertible codes in the merge regime are stable.

▶ **Lemma 14.** *All access-optimal linear MDS* $(n^I, k^I; n^F, k^F = \lambda k^I)$ *convertible codes are stable.*

Proof. In Appendix A.5. ◀

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 following key result.

▶ **Theorem 15.** *For all linear MDS* $(n^I, k^I; n^F, k^F = \lambda k^I)$ *convertible codes, the access cost of conversion is at least* $r^F + \lambda \min\{k^I, r^F\}$. *Furthermore, if* $r^I < r^F$, *the access cost of conversion is at least* $r^F + \lambda k^I$.

Proof. Follows from Theorem 12, Lemma 13, Lemma 14, and the definition of access cost. ◀

Next, in Section 4 we show that the lower bound of Theorem 15 is achievable for all parameters. Thus, Theorem 15 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 $r^F < k^I$.

▶ Remark 16. As discussed in Section 1, existing works [44, 59, 29] on code conversion model the problem within the framework of repair-efficient codes and optimize for network bandwidth consumed during conversion. These works require accessing every symbol, i.e. n^F symbols in total, during conversion. Our approach, thus, provides a significant reduction in terms of access cost in comparison to existing solutions for code conversion. To compare in terms of network bandwidth, consider each symbol to be a vector of size α as in these existing works. Then, our constructions (Section 4) can be easily implemented in a way that only requires a network bandwidth of $(\lambda - 1)r^F\alpha$ in the merge regime, by independently constructing each new symbol at the same location as one of the retired symbols it depends on. On the other hand, existing solutions [44, 59, 29] require network bandwidth of at least $[(\lambda - 1)k^I + r^F - r^I]\alpha$. Thus, although our constructions are optimized for access cost and not network bandwidth, they outperform existing solutions for several parameter regimes of practical interest, such as the merge regime with $r^I = r^F < k^I$ which corresponds to increasing the code rate for a code with rate greater than 0.5 (most storage systems use codes with rate greater than 0.5 and a conversion that increases the rate has been shown to be beneficial when a reduction in failure rate of storage devices is observed [32]).

4 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 Section 4.1, we describe the construction of the generator matrices for the initial and final code. Then, in Section 4.2, we describe sufficient conditions for optimality and show that this construction yields access-optimal convertible codes.

4.1 Explicit construction

Recall that, in the merge regime, $k^F = \lambda k^I$, for an integer $\lambda \geq 2$ and arbitrary n^I and n^F. 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 one can simply use the default approach. 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 and the degree D is a function of the convertible code parameters, which grows as $\Theta((n^F)^3)$ when $n^F > n^I$ and the rate of the final code is constant. 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 $[k^F] \times [r^F]$. Notice that this construction is stable, because it is access-optimal (recall Lemma 14). The unchanged symbols of the initial code are exactly the systematic symbols.

4.2 Proof of optimality

Throughout this section, we use the following notation for submatrices: let M be a $n \times m$ matrix, the submatrix of M defined by row indices $\{i_1, \ldots, i_a\}$ and column indices $\{j_1, \ldots, j_b\}$ 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 $\{i_1, \ldots, i_a\}$.

We first recall an important fact about systematic generator matrices of MDS codes.

▶ **Proposition 17** ([34]). *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, that is, every square submatrix of P is nonsingular*[3].

Thus, to be MDS, both $\mathbf{P}^I$ and $\mathbf{P}^F$ need to be superregular.

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 λk^I-dimensional vectors, where each initial codeword occupies a disjoint subset of k^I coordinates. To capture this property, we introduce the following definition.

[3] This definition of superregularity is stronger than the definition introduced in [19] in the context of convolutional codes.

▶ **Definition 18** (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 $i \in [\lambda]$, the submatrix $M_1[(i-1)n+1, \ldots, in; *]$ is t-column constructible from M_2.*

▶ **Theorem 19.** *A systematic $(n^I, k^I; n^F, k^F = \lambda k^I)$ 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 17 and the fact that $\mathbf{P}^F$ must be generated by accessing just r^F symbols from each initial codeword (Lemma 10). ◀

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 19.

We first show that the construction specified in Section 4.1 satisfies condition (1) of Theorem 19.

▶ **Lemma 20.** *Let $\mathbf{P}^I, \mathbf{P}^F$ be as defined in Section 4.1. Then $\mathbf{P}^F$ is r^F-column block-constructible from $\mathbf{P}^I$.*

Proof sketch. Given the structure of $\mathbf{P}^I$ and $\mathbf{P}^F$, it is easy to see that $\mathbf{P}^F$ can be obtained by "vertically stacking" copies of $\mathbf{P}^I$, with their columns appropriately scaled by powers of θ. Please refer to Appendix B.1 for full proof. ◀

It only remains to show that the construction in Section 4.1 satisfies condition (2) of Theorem 19, that is, that $\mathbf{P}^I$ and $\mathbf{P}^F$ are superregular.

▶ **Lemma 21.** *Let $\mathbf{P}^I, \mathbf{P}^F$ be as defined in Section 4.1. Then $\mathbf{P}^I$ and $\mathbf{P}^F$ are superregular, for sufficiently large field size.*

Proof sketch. Consider the minors of $\mathbf{P}^I$ and $\mathbf{P}^F$ as polynomials on θ. Due to the structure of the the matrices $\mathbf{P}^I$ and $\mathbf{P}^F$ as specified in Section 4.1, all of these are non-zero polynomials which cannot have θ as a root as long as the degree of the extension field is large enough. Therefore none of the minors can be zero. Please refer to Appendix B.2 for the full proof. ◀

Combining the above results leads to the following key result on the achievability of the lower bounds on access cost derived in Section 3.

▶ **Theorem 22.** *The explicit construction provided in Section 4.1 yields access-optimal linear MDS convertible codes for all parameter values in the merge regime.*

Proof. Follows from Theorem 19, Lemma 20, and Lemma 21. ◀

The construction presented in this section is practical only for very small values of these parameters since the required field size grows exponentially with the lengths of the initial and final codes. In Section 5 we present practical low-field-size constructions.

5 Low field-size convertible codes based on superregular Hankel arrays

In this section we present alternative constructions for $(n^I, k^I; n^F, k^F = \lambda k^I)$ convertible codes that require a significantly lower (polynomial) field size than the construction presented in Section 4. 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:

$$T_m: \quad \begin{matrix} b_1 & b_2 & b_3 & \cdots & b_{m-1} & b_m \\ b_2 & b_3 & \cdots & \cdots & b_m & \\ b_3 & \vdots & \iddots & \iddots & & \\ \vdots & \vdots & \iddots & & & \\ b_{m-1} & b_m & & & & \\ b_m & & & & & \end{matrix} \tag{1}$$

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 [52] (where the algorithm was employed to generate generalized Cauchy matrices to construct generalized Reed-Solomon codes). This algorithm is described in Appendix D for reference, although it is not necessary for understanding the constructions in this section.

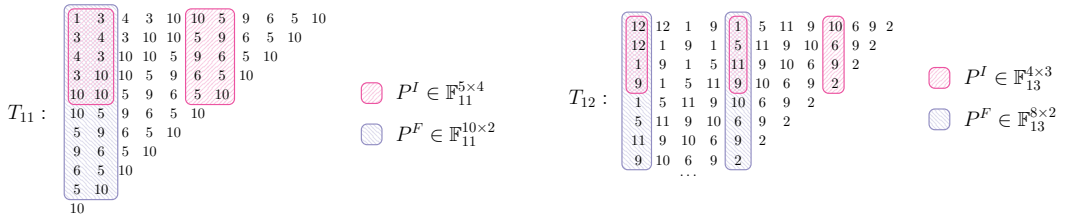
We construct the initial and final codes by taking submatrices $\mathbf{P}^I$ and $\mathbf{P}^F$ in a specific manner from superregular Hankel arrays (the submatrices have to be contained in the triangle where the array is defined). This guarantees that $\mathbf{P}^I$ and $\mathbf{P}^F$ are superregular. 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 subsection are (punctured) *generalized doubly-extended Reed-Solomon* codes [52].

The above idea yields a sequence of constructions with a tradeoff between the field size and the range 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 23, can be applied whenever $r^F \leq \lfloor r^I/\lambda \rfloor$, and requires a field size of $q \geq \max\{n^I, n^F\} - 1$. Construction Hankel-II, shown in Example 24, can be applied whenever $r^F \leq r^I - \lambda + 1$, and requires a field size of $q \geq k^I r^I$.

We then describe in detail the sequence of constructions that define a tradeoff between field size and coverage of r^F values in Section 5.1. In Section 5.2, we finalize with a discussion on the ability of these constructions to be optimal even when parameters of the final code are a priori unknown. Throughout this section we will assume that $\lambda \leq r^I \leq k^I$. The ideas presented here are still applicable when $r^I > k^I$, but the constructions and analysis change in minor ways.

▶ **Example 23** (Hankel-I)**.** Consider the parameters $(n^I = 9, k^I = 5; n^F = 12, k^F = 10)$ and the field $\mathbb{F}_{11}$. Notice that these parameters satisfy:

$$r^F = 2 \leq \left\lfloor \frac{r^I}{\lambda} \right\rfloor = 2 \quad \text{and} \quad q = 11 \geq \max\{n^I, n^F\} - 1 = 11.$$



(a) Hankel-I **(b)** Hankel-II

Figure 2 Examples of constructions based on Hankel arrays: (a) Hankel-I construction parity generator matrices for systematic $(n^I = 9, k^I = 5; n^F = 12, k^F = 10)$ 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 $(n^I = 7, k^I = 4; n^F = 10, k^F = 8)$ 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 $\mathbf{P}^I$.

First, construct a superregular Hankel array of size $n^F - 1 = 11$, T_{11}, employing the algorithm in [52]. Then, divide the $r^I = 4$ initial parities into $\lambda = 2$ 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 2a. 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 2a. Checking that these matrices are superregular follows from the superregularity of T_{11}. It is to check that both these matrices are superregular, which follows from the the superregularity of T_{11}. Furthermore, notice that the chosen parity matrices have the following structure:

$$\mathbf{P}^I = \begin{bmatrix} | & | & | & | \\ \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_3 & \mathbf{p}_4 \\ | & | & | & | \end{bmatrix}, \qquad \mathbf{P}^F = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 \\ \mathbf{p}_3 & \mathbf{p}_4 \end{bmatrix}.$$

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 19, and define an access-optimal convertible code.

▶ **Example 24** (Hankel-II). Consider parameters $(n^I = 7, k^I = 4; n^F = 10, k^F = 8)$ and field $\mathbb{F}_{13}$. Notice that these parameters satisfy:

$$r^F = 2 \leq r^I - \lambda + 1 = 2 \quad \text{and} \quad q = 13 \geq k^I r^I = 12$$

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 [52]. The submatrix $\mathbf{P}^I \in \mathbb{F}_{13}^{4\times 3}$ is formed by the top $k^I = 4$ rows and columns $1, k^I + 1 = 5$ and $2k^I + 1 = 9$ of T_{12}, as shown in Figure 2b. The submatrix $\mathbf{P}^F \in \mathbb{F}_{13}^{8\times 2}$ is formed by the top $k^F = 8$ rows and columns 1 and $k^I + 1 = 5$ of T_{12}, as shown in Figure 2b. 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:

$$\mathbf{P}^I = \begin{bmatrix} | & | & | \\ \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_3 \\ | & | & | \end{bmatrix}, \qquad \mathbf{P}^F = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 \\ \mathbf{p}_2 & \mathbf{p}_3 \end{bmatrix}$$

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 19, and define an access-optimal convertible code.

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

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

Therefore, Hankel-I, from Example 23 corresponds to Hankel_λ and Hankel-II from Example 24 corresponds to Hankel_{r^I}.

Construction of Hankel_s. Assume, for the sake of simplicity, that $s \mid r^I$ 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:

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

Without loss of generality, we consider $r^F = (s-\lambda+1)t$ (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 (1), 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 $\text{col}(S_i) = \{(i-1)k^I+1, (i-1)k^I+2, \ldots, (i-1)k^I+t\}$ of T_m. Matrix $\mathbf{P}^I$ is the submatrix formed by the top k^I rows and the columns indexed by the set $\text{col}(S_1) \cup \cdots \cup \text{col}(S_s)$ of T_m. Matrix $\mathbf{P}^F$ is the submatrix formed by the top λk^I rows and the columns indexed by the set $\text{col}(S_1) \cup \cdots \cup \text{col}(S_{s-\lambda+1})$ of T_m. This results in the following matrices $\mathbf{P}^I$ and $\mathbf{P}^F$:

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

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

▶ **Theorem 25.** *Given parameters k^I, r^I, λ, and a field $\mathbb{F}_q$ Hankel_s ($s \in \{\lambda, \ldots, r^I\}$) constructs an access-optimal $(n^I, k^I; n^F, k^F = \lambda k^I)$ convertible code if:*

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

Proof. In Appendix C.1. ◀

(Access-optimal) conversion process. During conversion, the k^I data symbols from each of the λ initial codewords remain unchanged, and become the $k^F = \lambda k^I$ 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 $i \in [\lambda]$, and then sum the λ 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 $l \in [r^F]$.

5.2 Handling a priori unknown parameters

So far, we had assumed that the parameters of the final code, n^F, k^F, are known a priori and are fixed. As discussed in Section 2, this is useful in developing an understanding of the fundamental limits of code conversion. When realizing code conversion in practice, however, the parameters n^F, k^F might not be known at code construction time (as it depends on the empirically observed failure rates). Thus, it is of interest to be able to *convert a code optimally to multiple different parameters*. The Hankel-array based constructions presented above indeed provide such a flexibility. Our constructions continue to enable access-optimal conversion for any $k^{F'} = \lambda' k^I$ and $n^{F'} = r^{F'} + k^{F'}$ with $0 \leq r^{F'} \leq r^F$ and $2 \leq \lambda' \leq \lambda$.

6 Related work

MDS erasure codes, such as Reed-Solomon codes, are widely used in storage systems because they achieve the optimal tradeoff between failure tolerance and storage overhead [43, 42]. However, the use of erasure codes in storage systems raises a host of other challenges. Several works in the literature have studied these aspects.

The encoding and decoding of data, and the finite field arithmetic that they require, can be compute intensive. Motivated by this, *array codes* [7, 70, 31, 28] are designed to use XOR operations exclusively, which are typically faster to execute, and aim to decrease the complexity of encoding and decoding.

The repair of failed nodes can require a large amount of network bandwidth. Several approaches have been proposed to alleviate this problem. Dimakis et al. [16] 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 [16], 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 [10, 47, 55, 60, 68, 69, 64, 11, 40, 53, 71, 73, 50, 54, 22, 13, 35, 36], generalizations [56, 58, 1], and more efficient repair algorithms for Reed-Solomon codes [57, 26, 72, 15, 66, 37, 14, 67]. It has been shown that meeting the lower bound on the repair bandwidth requirement when MDS property and high rate are desired necessitates large sub-packetization [65, 23, 5, 3], which negatively affects certain key performance metrics in storage systems [45]. To overcome this issue, several works [49, 25] have proposed code constructions that relax the requirement of meeting lower bounds on IO and bandwidth requirements for repair operations. For example, the Piggybacking framework [49] provides a general framework to construct repair-efficient codes by transforming any existing codes, while allowing a small sub-packetization (even as small as 2).

Several works [46, 39] study the problem of two stage encoding: first generating a certain number of parities during the encoding process and then adding additional parities. As discussed in [46], adding additional parities can be conceptually viewed as a repair process by considering the new parity nodes to be generated as failed nodes. Furthermore, as shown in [55], for MDS codes, the bandwidth requirement for *repair of even a single node* is lower bounded by the same amount as in regenerating codes that require repair of *all* nodes. Thus one can always employ a regenerating code to add additional parities with minimum

bandwidth overhead. However, when MDS property and high rate are desired, as discussed above, using regenerating codes requires a large sub-packetization. The paper [39] employs the Piggybacking framework [48, 49] to construct codes that reduce the sub-packetization factor for two-stage encoding. The scenario of adding a fixed number of additional parities, when viewed under the setting of conversions, corresponds to having $k^I = k^F$ and $n^I < n^F$.

Existing works that consider changing the parameters n and k of already encoded data [44, 59, 29] consider a model similar to the regenerating codes model, wherein symbols are represented as vectors and communication is modeled via an information flow graph. In order to accommodate the changes in parameter k, the dimension α of each vector is changed, and constructions exploit repair efficiency to achieve conversion efficiency. However, this approach has the disadvantage that during conversion every symbol must be accessed. Our approach circumvents this problem by considering multiple codewords at a time, which allows convertible codes to achieve significantly smaller access cost during conversion. For a more detailed comparison between convertible codes and existing works on code conversion, see Remark 16 in Section 3.

Another class of codes, called *locally repairable codes* (LRCs) [21, 51, 8, 20, 41, 61, 33, 12, 63, 62, 6, 17, 2, 38, 27], focuses 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 and degraded read performance, since missing information can be recovered by accessing a small subset of symbols. The objective of LRCs and convertible codes optimized for access cost is similar, as both aim to minimize the number of symbols that need to be accessed for different operations in storage systems.

7 Conclusions and future directions

In this paper, we propose a new framework to model the code conversion problem, that of converting data encoded with an $[n^I, k^I]$ code into data encoded with an $[n^F, k^F]$ code in a resource-efficient manner. The code conversion problem is motivated by the practical necessity of reducing the overhead of redundancy adaptation in erasure-coded storage systems [32]. We present the framework of convertible codes for studying code conversions, and fully characterize the fundamental limits on the access cost of conversions for an important regime of convertible codes. Furthermore, we present practical low-field-size constructions for access-optimal convertible codes for a wide range of parameters.

This work leads to a number of challenging and potentially impactful open problems. An important future direction is to go beyond the merge regime considered in this paper. While the construction techniques presented in this paper can be easily extended to upper bound the access cost of some parameter values outside the merge regime, identifying the fundamental limits on the access cost in general and constructing access-optimal convertible codes for all parameters remains open. Another important future direction is to analyze the fundamental limits of convertible codes on the overhead of other resources, such as disk IO (i.e., device bandwidth), communication (network bandwidth), computation (CPU), and construct convertible codes optimizing these resources. Note that while the access-optimal convertible codes considered in this paper also reduce the total disk IO, communication and computation during conversion as compared to the default approach, the overhead on these other resources may not be optimal. A final important direction is to explore additional applications of convertible codes beyond our initial motivation, to other problems within theoretical computer science where codes are used.

References

1 Vitaly Abdrashitov, N Prakash, and Muriel Médard. The storage vs repair bandwidth trade-off for multiple failures in clustered storage networks. In *2017 IEEE Information Theory Workshop (ITW)*, pages 46–50. IEEE, 2017.

2 Abhishek Agarwal, Alexander Barg, Sihuang Hu, Arya Mazumdar, and Itzhak Tamo. Combinatorial alphabet-dependent bounds for locally recoverable codes. *IEEE Transactions on Information Theory*, 64(5):3481–3492, 2018.

3 Omar Alrabiah and Venkatesan Guruswami. An Exponential Lower Bound on the Sub-packetization of MSR Codes. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, pages 979–985, New York, NY, USA, 2019. ACM.

4 Apache Software Foundation. Apache hadoop: HDFS erasure coding. Accessed: 2019-07-23. URL: `https://hadoop.apache.org/docs/r3.0.0/hadoop-project-dist/hadoop-hdfs/HDFSErasureCoding.html`.

5 SB Balaji and P Vijay Kumar. A tight lower bound on the sub-packetization level of optimal-access MSR and MDS codes. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 2381–2385. IEEE, 2018.

6 Alexander Barg, Kathryn Haymaker, Everett W Howe, Gretchen L Matthews, and Anthony Várilly-Alvarado. Locally recoverable codes from algebraic curves and surfaces. In *Algebraic Geometry for Coding Theory and Cryptography*, pages 95–127. Springer, 2017.

7 M. Blaum, J. Brady, J. Bruck, and Jai Menon. EVENODD: an efficient scheme for tolerating double disk failures in RAID architectures. *IEEE Transactions on Computers*, 44(2):192–202, February 1995.

8 Mario Blaum, James Lee Hafner, and Steven Hetzler. Partial-MDS codes and their application to RAID type of architectures. *IEEE Transactions on Information Theory*, 59(7):4510–4519, 2013.

9 Dhruba Borthakur, Rodrigo Schmidt, Ramkumar Vadali, Scott Chen, and Patrick Kling. HDFS RAID - Facebook. URL: `http://www.slideshare.net/ydn/hdfs-raid-facebook`.

10 Viveck R Cadambe, Cheng Huang, Jin Li, and Sanjeev Mehrotra. Polynomial length MDS codes with optimal repair in distributed storage. In *2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, pages 1850–1854. IEEE, 2011.

11 Viveck R Cadambe, Syed A Jafar, Hamed Maleki, Kannan Ramchandran, and Changho Suh. Asymptotic interference alignment for optimal repair of MDS codes in distributed storage. *IEEE Transactions on Information Theory*, 59(5):2974–2987, 2013.

12 Viveck R Cadambe and Arya Mazumdar. Bounds on the size of locally recoverable codes. *IEEE Transactions on Information Theory*, 61(11):5787–5794, 2015.

13 Ameera Chowdhury and Alexander Vardy. New Constructions of MDS Codes with Asymptotically Optimal Repair. In *2018 IEEE International Symposium on Information Theory*, pages 1944–1948, 2018.

14 Hoang Dau, Iwan M Duursma, Han Mao Kiah, and Olgica Milenkovic. Repairing Reed-Solomon codes with multiple erasures. *IEEE Transactions on Information Theory*, 64(10):6567–6582, 2018.

15 Hoang Dau and Olgica Milenkovic. Optimal repair schemes for some families of full-length Reed-Solomon codes. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 346–350. IEEE, 2017.

16 A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran. Network Coding for Distributed Storage Systems. *IEEE Transactions on Information Theory*, 56(9):4539–4551, September 2010.

17 S Luna Frank-Fischer, Venkatesan Guruswami, and Mary Wootters. Locality via Partially Lifted Codes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

18 S. Ghemawat, H. Gobioff, and S.T. Leung. The Google file system. In *ACM SIGOPS Operating Systems Review*, volume 37-5, pages 29–43. ACM, 2003.

19 H. Gluesing-Luerssen, J. Rosenthal, and R. Smarandache. Strongly-MDS convolutional codes. *IEEE Transactions on Information Theory*, 52(2):584–598, February 2006.

20 Parikshit Gopalan, Cheng Huang, Bob Jenkins, and Sergey Yekhanin. Explicit maximally recoverable codes with locality. *IEEE Transactions on Information Theory*, 60(9):5245–5256, 2014.

21 Parikshit Gopalan, Cheng Huang, Huseyin Simitci, and Sergey Yekhanin. On the locality of codeword symbols. *IEEE Transactions on Information Theory*, 58(11):6925–6934, 2012.

22 Sreechakra Goparaju, Arman Fazeli, and Alexander Vardy. Minimum Storage Regenerating Codes for All Parameters. *IEEE Transactions on Information Theory*, 63(10):6318–6328, 2017.

23 Sreechakra Goparaju, Itzhak Tamo, and Robert Calderbank. An improved sub-packetization bound for minimum storage regenerating codes. *IEEE Transactions on Information Theory*, 60(5):2770–2779, 2014.

24 Sivakanth Gopi, Venkatesan Guruswami, and Sergey Yekhanin. Maximally Recoverable LRCs: A field size lower bound and constructions for few heavy parities. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2154–2170. SIAM, 2019.

25 Venkatesan Guruswami and Ankit Singh Rawat. MDS code constructions with small sub-packetization and near-optimal repair bandwidth. In *ACM-SIAM Symposium on Discrete Algorithms*, 2017.

26 Venkatesan Guruswami and Mary Wootters. Repairing Reed-Solomon codes. In *ACM Symposium on Theory of Computing*, pages 216–226, 2016.

27 Venkatesan Guruswami, Chaoping Xing, and Chen Yuan. How Long Can Optimal Locally Repairable Codes Be? In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

28 James Lee Hafner. WEAVER codes: Highly fault tolerant erasure codes for storage systems. In *Proceedings of the 4th Conference on USENIX Conference on File and Storage Technologies - Volume 4*, FAST'05, pages 16–16, Berkeley, CA, USA, 2005. USENIX Association.

29 Yuchong Hu, Xiaoyang Zhang, Patrick PC Lee, and Pan Zhou. Generalized Optimal Storage Scaling via Network Coding. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 956–960. IEEE, 2018.

30 C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin. Erasure Coding in Windows Azure Storage. In *Proceedings of USENIX Annual Technical Conference (ATC)*, 2012.

31 Cheng Huang and Lihao Xu. STAR: An efficient coding scheme for correcting triple storage node failures. *IEEE Transactions on Computers*, 57(7):889–901, 2008.

32 Saurabh Kadekodi, K. V. Rashmi, and Gregory R. Ganger. Cluster storage systems gotta have HeART: improving storage efficiency by exploiting disk-reliability heterogeneity. *USENIX FAST*, 2019.

33 Govinda M Kamath, N Prakash, V Lalitha, and P Vijay Kumar. Codes with local regeneration and erasure correction. *IEEE Transactions on Information Theory*, 60(8):4637–4660, 2014.

34 F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-holland Publishing Company, 2nd edition, 1978.

35 Kaveh Mahdaviani, Ashish Khisti, and Soheil Mohajer. Bandwidth adaptive & error resilient MBR exact repair regenerating codes. *IEEE Transactions on Information Theory*, 65(5):2736–2759, 2018.

36 Kaveh Mahdaviani, Soheil Mohajer, and Ashish Khisti. Product matrix MSR codes with bandwidth adaptive exact repair. *IEEE Transactions on Information Theory*, 64(4):3121–3135, 2018.

37 Jay Mardia, Burak Bartan, and Mary Wootters. Repairing multiple failures for scalar MDS codes. *IEEE Transactions on Information Theory*, 65(5):2661–2672, 2018.

38 A. Mazumdar. Capacity of Locally Recoverable Codes. In *2018 IEEE Information Theory Workshop*, pages 1–5, November 2018.

39 S. Mousavi, T. Zhou, and C. Tian. Delayed Parity Generation in MDS Storage Codes. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 1889–1893, June 2018.

40 D Papailiopoulos, A Dimakis, and V Cadambe. Repair Optimal Erasure Codes through Hadamard Designs. *IEEE Transactions on Information Theory*, 59(5):3021–3037, May 2013.

41 Dimitris S Papailiopoulos and Alexandros G Dimakis. Locally repairable codes. *IEEE Transactions on Information Theory*, 60(10):5843–5855, 2014.

42 David A Patterson, Garth Gibson, and Randy H Katz. *A case for redundant arrays of inexpensive disks (RAID)*, volume 17-3. ACM, 1988.

43 J.S. Plank. T1: Erasure codes for storage applications. *Proceedings of the 4th USENIX Conference on File and Storage Technologies*, pages 1–74, January 2005.

44 Brijesh Kumar Rai, Vommi Dhoorjati, Lokesh Saini, and Amit K Jha. On adaptive distributed storage systems. In *2015 IEEE international symposium on information theory (ISIT)*, pages 1482–1486. IEEE, 2015.

45 K. V. Rashmi, Nihar B Shah, Dikang Gu, Hairong Kuang, Dhruba Borthakur, and Kannan Ramchandran. A Hitchhiker's guide to fast and efficient data reconstruction in erasure-coded data centers. In *ACM SIGCOMM*, 2014.

46 K. V. Rashmi, Nihar B Shah, and P Vijay Kumar. Enabling node repair in any erasure code for distributed storage. In *2011 IEEE International Symposium on Information Theory Proceedings*, pages 1235–1239. IEEE, 2011.

47 K. V. Rashmi, Nihar B Shah, and P Vijay Kumar. Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction. *IEEE Transactions on Information Theory*, 57(8):5227–5239, 2011.

48 K. V. Rashmi, Nihar B Shah, and Kannan Ramchandran. A piggybacking design framework for read-and download-efficient distributed storage codes. In *2013 IEEE International Symposium on Information Theory*, 2013.

49 K. V. Rashmi, Nihar B Shah, and Kannan Ramchandran. A piggybacking design framework for read-and download-efficient distributed storage codes. *IEEE Transactions on Information Theory*, 63(9):5802–5820, 2017.

50 Ankit Singh Rawat, O Ozan Koyluoglu, and Sriram Vishwanath. Progress on high-rate MSR codes: Enabling arbitrary number of helper nodes. In *2016 Information Theory and Applications Workshop (ITA)*, pages 1–6. IEEE, 2016.

51 Ankit Singh Rawat, Onur Ozan Koyluoglu, Natalia Silberstein, and Sriram Vishwanath. Optimal locally repairable and secure codes for distributed storage systems. *IEEE Transactions on Information Theory*, 60(1):212–236, 2013.

52 Ron M Roth and Gadiel Seroussi. On Generator Matrices of MDS Codes. *IEEE Transactions on Infortmation Theory*, 31(6):826–830, November 1985.

53 Birenjith Sasidharan, Gaurav Kumar Agarwal, and P Vijay Kumar. A high-rate MSR code with polynomial sub-packetization level. In *2015 IEEE International Symposium on Information Theory (ISIT)*, pages 2051–2055. IEEE, 2015.

54 Birenjith Sasidharan, Myna Vajha, and P Vijay 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 *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 2048–2052. IEEE, 2017.

55 Nihar B Shah, K. V. Rashmi, P Vijay Kumar, and Kannan Ramchandran. Interference alignment in regenerating codes for distributed storage: Necessity and code constructions. *IEEE Transactions on Information Theory*, 58(4):2134–2158, 2011.

56 Nihar B Shah, KV Rashmi, and P Vijay Kumar. A flexible class of regenerating codes for distributed storage. In *2010 IEEE International Symposium on Information Theory*, pages 1943–1947. IEEE, 2010.

57 Karthikeyan Shanmugam, Dimitris S Papailiopoulos, Alexandros G Dimakis, and Giuseppe Caire. A repair framework for scalar MDS codes. *IEEE Journal on Selected Areas in Communications*, 32(5):998–1007, 2014.

58 Kenneth W Shum. Cooperative regenerating codes for distributed storage systems. In *2011 IEEE International Conference on Communications (ICC)*, pages 1–5. IEEE, 2011.

59 Mridupawan Sonowal and Brijesh Kumar Rai. On adaptive distributed storage systems based on functional MSR code. In *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pages 338–343. IEEE, 2017.

60 C. Suh and Kannan Ramchandran. Exact-Repair MDS Code Construction Using Interference Alignment. *IEEE Transactions on Information Theory*, pages 1425–1442, March 2011.

61 Itzhak Tamo and Alexander Barg. A family of optimal locally recoverable codes. *IEEE Transactions on Information Theory*, 60(8):4661–4676, 2014.

62 Itzhak Tamo, Alexander Barg, and Alexey Frolov. Bounds on the parameters of locally recoverable codes. *IEEE Transactions on Information Theory*, 62(6):3070–3083, 2016.

63 Itzhak Tamo, Dimitris S Papailiopoulos, and Alexandros G Dimakis. Optimal locally repairable codes and connections to matroid theory. *IEEE Transactions on Information Theory*, 62(12):6661–6671, 2016.

64 Itzhak Tamo, Zhiying Wang, and Jehoshua Bruck. Zigzag codes: MDS array codes with optimal rebuilding. *IEEE Transactions on Information Theory*, 59(3):1597–1616, 2013.

65 Itzhak Tamo, Zhiying Wang, and Jehoshua Bruck. Access versus bandwidth in codes for storage. *IEEE Transactions on Information Theory*, 60(4):2028–2037, 2014.

66 Itzhak Tamo, Min Ye, and Alexander Barg. Optimal repair of Reed-Solomon codes: Achieving the cut-set bound. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 216–227. IEEE, 2017.

67 Itzhak Tamo, Min Ye, and Alexander Barg. The Repair Problem for Reed–Solomon Codes: Optimal Repair of Single and Multiple Erasures With Almost Optimal Node Size. *IEEE Transactions on Information Theory*, 65(5):2673–2695, 2018.

68 Zhiying Wang, Itzhak Tamo, and Jehoshua Bruck. On codes for optimal rebuilding access. In *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1374–1381. IEEE, 2011.

69 Zhiying Wang, Itzhak Tamo, and Jehoshua Bruck. Long MDS codes for optimal repair bandwidth. In *2012 IEEE International Symposium on Information Theory Proceedings*, pages 1182–1186. IEEE, 2012.

70 Lihao Xu and Jehoshua Bruck. X-code: MDS array codes with optimal encoding. *IEEE Transactions on Information Theory*, 45(1):272–276, 1999.

71 M. Ye and A. Barg. Explicit Constructions of Optimal-Access MDS Codes With Nearly Optimal Sub-Packetization. *IEEE Transactions on Information Theory*, 63(10):6307–6317, October 2017.

72 Min Ye and Alexander Barg. Explicit constructions of MDS array codes and RS codes with optimal repair bandwidth. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 1202–1206. IEEE, 2016.

73 Min Ye and Alexander Barg. Explicit constructions of high-rate MDS array codes with optimal repair bandwidth. *IEEE Transactions on Information Theory*, 63(4):2001–2014, 2017.

A Proofs of Section 3

A.1 Notation used in proofs

Let $\mathcal{C}^I$ be an $[n^I, k^I]$ MDS code over field $\mathbb{F}_q$, specified by generator matrix $\mathbf{G}^I$, with columns (that is, encoding vectors) $\{\mathbf{g}^I_1, \ldots, \mathbf{g}^I_{n^I}\} \subseteq \mathbb{F}_q^{k^I}$. Let $\lambda \geq 2$ be an integer, and let $\mathcal{C}^F$ be an $[n^F, k^F = \lambda k^I]$ MDS code over field $\mathbb{F}_q$, specified by generator matrix $\mathbf{G}^F$, with columns (that is, encoding vectors) $\{\mathbf{g}^F_1, \ldots, \mathbf{g}^F_{n^F}\} \subseteq \mathbb{F}_q^{k^F}$. Let $r^I = n^I - k^I$ and $r^F = n^F - k^F$. When $\mathcal{C}^I$ and $\mathcal{C}^F$ are systematic, r^I and r^F correspond to the initial number of parities and final number of parities, respectively. All vectors are assumed to be column vectors. We will use the notation $\mathbf{v}[l]$ to denote the l-th coordinate of a vector $\mathbf{v}$.

We will represent *all* the code symbols in the initial codewords as being generated by a single $\lambda k^I \times \lambda n^I$ matrix $\tilde{\mathbf{G}}^I$, with encoding vectors $\{\tilde{\mathbf{g}}^I_{i,j} \mid i \in [\lambda], j \in [n^I]\} \subseteq \mathbb{F}_q^{k^F}$. This representation can be viewed as embedding the column vectors of the generator matrix $\mathbf{G}^I$ in an λk^I-dimensional space, where the index set $\mathcal{K}_i = \{(i-1)k^I+1, \ldots, ik^I\}, i \in [\lambda]$ corresponds to the encoding vectors for initial codeword i. Let $\tilde{\mathbf{g}}^I_{i,j}$ denote the j-th encoding vector in the initial codeword i in this (embedded) representation. Thus, $\tilde{\mathbf{g}}^I_{i,j}[l] = \mathbf{g}^I_j[l - (i-1)k^I]$ for $l \in \mathcal{K}_i$, and $\tilde{\mathbf{g}}^I_{i,j}[l] = 0$ otherwise. As an example, Figure 3 shows the values of the defined terms for the single parity-check code from Figure 1 with $n^I = 3, k^I = 2, n^F = 5, k^F = 4$.

At times, focus will be only on the coordinates of an encoding vector of a certain initial codeword i. For this purpose, define $\text{proj}_{\mathcal{K}_i}(\mathbf{v}) \in \mathbb{F}_q^{k^I}$ to be the projection of $\mathbf{v} \in \mathbb{F}_q^{k^F}$ to the coordinates in an index set $\mathcal{K}_i$, and for a set $\mathcal{V}$ of vectors, $\text{proj}_{\mathcal{K}_i}(\mathcal{V}) = \{\text{proj}_{\mathcal{K}_i}(\mathbf{v}) \mid \mathbf{v} \in \mathcal{V}\}$. For example, $\text{proj}_{\mathcal{K}_i}(\tilde{\mathbf{g}}^I_{i,j}) = \mathbf{g}^I_j$ for all $i \in [\lambda]$ and $j \in [n^I]$.

The following sets of vectors are defined: the encoding vectors from initial codeword i, $\mathcal{S}^I_i = \{\tilde{\mathbf{g}}^I_{i,j} \mid j \in [n^I]\}$, all the encoding vectors from all the initial codewords, $\mathcal{S}^I = \cup_{i \in [\lambda]} \mathcal{S}^I_i$, and all the encoding vectors from the final codeword $\mathcal{S}^F = \{\mathbf{g}^F_j \mid j \in [n^F]\}$.

We use the term *unchanged symbols* to refer to symbols from the initial codewords that remain as is (that is, unchanged) in the final codeword. The symbols in the final codeword that were not present in the initial codewords are called *new*, and the symbols from the initial codewords that do not carry over to the final codeword are called *retired*. For example, in Figure 1, all the data symbols are unchanged symbols (unshaded boxes), the single parity symbol of the final codeword is a new symbol, and the two parity blocks from the initial codewords are retired symbols. Each unchanged symbol corresponds to a pair of identical initial and final encoding vectors, that is, a tuple of indices (i, j, l) such that $\tilde{\mathbf{g}}^I_{i,j} = \mathbf{g}^F_l$. For instance, the example in Figure 1 has four unchanged symbols, corresponding to the identical encoding vectors $\tilde{\mathbf{g}}^I_{i,j} = \mathbf{g}^F_{2(i-1)+j}$ for $i, j \in [2]$. The final encoding vectors $\mathcal{S}^F$ can thus be partitioned into the following sets: *unchanged encoding vectors* from initial codeword i, $\mathcal{U}_i = \mathcal{S}^F \cap \mathcal{S}^I_i$ for all $i \in [\lambda]$, and *new encoding vectors* $\mathcal{N} = \mathcal{S}^F \setminus \mathcal{S}^I$.

From the point of view of conversion cost, unchanged symbols are ideal, because they require no extra work. On the other hand, constructing new symbols require accessing symbols from the initial codewords. When a symbol from the initial codewords is accessed, all of its contents are downloaded to a central location, where they are available for the construction of all new symbols. For example, in Figure 1, one symbol from each initial codeword is accessed during conversion.

During conversion, new symbols are constructed by reading symbols from the initial codewords. That is, every new encoding vector is simply a linear combination of a specific subset of $\mathcal{S}^I$. Define the *read access set* for an MDS $(n^I, k^I; n^F, k^F = \lambda k^I)$ convertible code as the set of tuples $\mathcal{D} \in [\lambda] \times [n^I]$ such that the set of new encoding vectors $\mathcal{N}$ is contained in the span of the set $\{\tilde{\mathbf{g}}^I_{i,j} \mid (i,j) \in \mathcal{D}\}$. Furthermore, define the index sets $\mathcal{D}_i = \{j \mid (i,j) \in \mathcal{D}\}$, $\forall i \in [\lambda]$ which denote the encoding vectors accessed from each initial codeword.

$$\mathbf{G}^I = \begin{matrix} \mathbf{g}_1^I & \mathbf{g}_2^I & \mathbf{g}_3^I \end{matrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \quad \tilde{\mathbf{G}}^I = \begin{matrix} \tilde{\mathbf{g}}_{1,1}^I & \tilde{\mathbf{g}}_{1,2}^I & \tilde{\mathbf{g}}_{1,3}^I & \tilde{\mathbf{g}}_{2,1}^I & \tilde{\mathbf{g}}_{2,2}^I & \tilde{\mathbf{g}}_{2,3}^I \end{matrix} \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \quad \mathbf{G}^F = \begin{matrix} \mathbf{g}_1^F & \mathbf{g}_2^F & \mathbf{g}_3^F & \mathbf{g}_4^F & \mathbf{g}_5^F \end{matrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Figure 3 Generator matrices for a specific $(n^I = 3, k^I = 2; n^F = 5, k^F = 4)$ convertible code: $\mathbf{G}^I$ is the generator matrix of the initial code; $\tilde{\mathbf{G}}^I$ is the generator matrix of all initial codewords; $\mathbf{G}^F$ is the generator matrix of the final code.

A.2 Proof of Lemma 10

The notation used in this proof is introduced in Appendix A.1. By the MDS property, every subset $\mathcal{V} \in \mathcal{S}^F$ of size at most $k^F = \lambda k^I$ is linearly independent. For any initial codeword $i \in [\lambda]$, consider the set of all unchanged encoding vectors from other codewords, $\cup_{\ell \neq i} \mathcal{U}_\ell$, and pick any subset of new encoding vectors $\mathcal{W} \subseteq \mathcal{N}$ of size $|\mathcal{W}| = \min\{k^I, r^F\}$. Consider the subset $\mathcal{V} = (\cup_{\ell \neq i} \mathcal{U}_\ell \cup \mathcal{W})$: it is true that $\mathcal{V} \subseteq \mathcal{S}^F$ and $|\mathcal{V}| = (\lambda - 1)k^I + \min\{k^I, r^F\} \leq k^F$. 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 $\mathcal{W}_i = \text{proj}_{\mathcal{K}_i}(\mathcal{W})$ has rank equal to $\min\{k^I, r^F\}$ (recall from Appendix A.1 that $\mathcal{K}_i$ is the set of coordinates belonging to initial codeword i). However, by definition, the subset $\mathcal{W}_i$ must be contained in the span of $\{\mathbf{g}_j^I \mid j \in \mathcal{D}_i\}$. Therefore, the rank of $\{\mathbf{g}_j^I \mid j \in \mathcal{D}_i\}$ is at least that of $\mathcal{W}_i$, which implies that $|\mathcal{D}_i| \geq \min\{k^I, r^F\}$. ◀

A.3 Proof of Lemma 11

The notation used in this proof is introduced in Appendix A.1. When $r^F \geq k^I$, this lemma is equivalent to Lemma 10, so assume $r^I < r^F < k^I$. From the proof of Lemma 10, for every initial codeword $i \in [\lambda]$ 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 $k^F = \lambda k^I$ 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, $\mathcal{W}_1 = (\{\tilde{\mathbf{g}}_{i,j}^I \mid j \in \mathcal{D}_i\} \cap \mathcal{U}_i)$. For the remaining vectors (if any), choose an arbitrary subset of new encoding vectors, $\mathcal{W}_2 \subseteq \mathcal{N}$, such that:

$$|\mathcal{W}_2| = \min\{k^I - |\mathcal{W}_1|, r^F\}. \tag{2}$$

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 $k^F = \lambda k^I$, 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 $\widetilde{\mathcal{W}} = \text{proj}_{\mathcal{K}_i}(\mathcal{W}_1) \cup \text{proj}_{\mathcal{K}_i}(\mathcal{W}_2)$ has rank $|\mathcal{W}_1| + |\mathcal{W}_2|$. Moreover, $\widetilde{\mathcal{W}}$ is contained in the span of $\{\mathbf{g}_j^I \mid j \in \mathcal{D}_i\}$ by definition, so it holds that:

$$|\mathcal{D}_i| \geq |\mathcal{W}_1| + |\mathcal{W}_2|. \tag{3}$$

From Equation (2), 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 (3) 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 (3) it holds that:

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

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:

$$|\mathcal{D}_i| \leq |\mathcal{W}_1| + r^I. \tag{5}$$

By combining Equation (4) and Equation (5), 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 always holds, and $|\mathcal{D}_i| \geq k$. ◀

A.4 Proof of Lemma 13

The notation used in this proof is introduced in Appendix A.1. 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 12 still hold.

Case 1: $r^I \geq r^F$. Let $i \in [\lambda]$ 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 $|\mathcal{V}| = \lambda k^I$ 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 $\text{proj}_{\mathcal{K}_i}(\mathcal{V})$ 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 10 continues to hold for non-stable codes.

Case 2: $r^I < r^F$. Let $i \in [\lambda]$ be an arbitrary initial codeword, let $\mathcal{W}_1 = (\{\tilde{\mathbf{g}}^I_{i,j} \mid j \in \mathcal{D}_i\} \cap \mathcal{U}_i)$ 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 $k^F = \lambda k^I$ encoding vectors from the final codeword such that $\mathcal{W}_1 \subseteq \mathcal{V}$ and the size of the intersection $\mathcal{W}_3 = (S \cap \mathcal{W}_2)$ is minimized. Since $\mathcal{V}$ may exclude at most r^F encoding vectors from the final codeword, it holds that:

$$|\mathcal{W}_3| = \max\{0, |\mathcal{W}_2| - r^F\}. \tag{6}$$

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:

$$|\mathcal{D}_i| + |\mathcal{W}_3| \geq k^I. \tag{7}$$

From Equation (6), there are two cases:

Subcase 2.1: $|\mathcal{W}_2| - r^F \leq 0$. Then $|\mathcal{W}_3| = 0$, and by Equation (7) it holds that $|\mathcal{D}_i| \geq k^I$, which matches the bound of Lemma 11.

Subcase 2.2: $|\mathcal{W}_2| - r^F > 0$. Then $|\mathcal{W}_3| = |\mathcal{W}_2| - r^F$, and by Equation (7) it holds that:

$$|\mathcal{D}_i| + |\mathcal{W}_2| - r^F \geq k^I. \tag{8}$$

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

$$|\mathcal{D}_i| + |\mathcal{U}_i| - |\mathcal{W}_1| \leq k^I + r^I. \tag{9}$$

By using Equation (8), Equation (9) 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. ◀

A.5 Proof of Lemma 14

Lemma 13 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 4). 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 $(n^I, k^I; n^F, k^F = \lambda k^I)$ convertible code is strictly greater than the access cost of an access-optimal $(n^I, k^I; n^F, k^F = \lambda k^I)$ convertible code. ◀

B Proofs of Section 4

B.1 Proof of Lemma 20

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

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

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

B.2 Proof of Lemma 21

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:

$$\det(\mathbf{R}) = \sum_{\sigma \in \operatorname{Perm}(t)} \operatorname{sgn}(\sigma) \prod_{l=1}^{t} \mathbf{R}[l, \sigma(l)] = \sum_{\sigma \in \operatorname{Perm}(t)} \operatorname{sgn}(\sigma) \theta^{E_\sigma} \tag{10}$$

$$\text{where} \qquad E_\sigma = \sum_{l=1}^{t} (i_l - 1)(j_{\sigma(l)} - 1), \tag{11}$$

Perm(t) is the set of all permutations on t elements, and $\text{sgn}(\sigma) \in \{-1, 1\}$ 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 $\sigma^* \in \text{Perm}(t)$ for which E_{σ^*} achieves this value, and that this is the maximum over all permutations in Perm(t). This means that $f_{\mathbf{R}}$ has a leading term of degree E_{σ^*}.

To prove this, we show that any permutation $\sigma \in \text{Perm}(t)\backslash\{\sigma^*\}$ can be modified into a permutation σ' such that $E_{\sigma'} > E_{\sigma}$. Specifically, we show that $\sigma^* = \sigma_{\text{id}}$, the identity permutation. Consider $\sigma \in \text{Perm}(t)\backslash\{\sigma_{\text{id}}\}$: 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:

$$E_{\sigma'} - E_{\sigma} = (i_a - 1)(j_a - 1) + (i_b - 1)(j_c - 1) - (i_a - 1)(j_c - 1) - (i_b - 1)(j_a - 1) \quad (12)$$

$$= (i_b - i_a)(j_c - j_a) > 0 \quad (13)$$

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

Let $E^*(\lambda, k^I, r^I, r^F)$ be the maximum degree of $f_{\mathbf{R}}$ over all submatrices $\mathbf{R}$ of $\mathbf{P}^I$ or $\mathbf{P}^F$. Then, $E^*(\lambda, k^I, r^I, r^F)$ 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:

$$E^*(\lambda, k^I, r^I, r^F) = \max\left\{\sum_{i=0}^{r^F-1} i(\lambda k^I - r^F + i), \sum_{i=0}^{r^I-1} i(k^I - r^I + i), \sum_{i=0}^{k^I-1} i(r^I - k^I + i)\right\}$$

$$= (1/6) \cdot \max\left\{\begin{matrix} r^F(r^F - 1)(3\lambda k^I - r^F - 1), \\ r^I(r^I - 1)(3k^I - r^I - 1), \\ k^I(k^I - 1)(3r^I - k^I - 1) \end{matrix}\right\}.$$

Let $D = E^*(\lambda, k^I, r^I, r^F) + 1$. 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 $D > \deg(f_{\mathbf{R}})$ [34]. ◀

C Proofs of Section 5

C.1 Proof of Theorem 25

Consider the construction Hankel$_s$ described in this section, for some $s \in \{\lambda, \ldots, r^I\}$. 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 $l, l+t, \ldots, l+(\lambda-1)t$ 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 19 and hence the convertible code constructed by Hankel$_s$ is access-optimal. ◀

D Algorithm for constructing superregular Hankel triangular arrays

In this appendix we describe the algorithm from [52] for constructing a superregular Hankel triangular array over any finite field. This is supplied as reference and is not necessary for understanding the constructions described in this paper. We note that the algorithm outlined in [52] 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 $m \leq q$.

Let $\mathbb{F}_q$ be a given base field, and let $m \leq q$ be the size of the output triangular array T_m. The triangular array T_m has Hankel form, as shown in Equation (1). Therefore, it suffices to specify the entries $b_1, b_2, \ldots, b_m$ in the first column of T_m. On input $m \leq q$, 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 $\sigma_{-1}, \sigma_0, \ldots, \sigma_m \in \mathbb{F}_q$ 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 [52] for a proof of this fact.