Locally Decodable Codes
Locally Decodable Codes

Sergey Yekhanin

Microsoft Research Silicon Valley
Mountain View, CA 94043
USA
yekhanin@microsoft.com
Editorial Scope

Foundations and Trends® in Theoretical Computer Science will publish survey and tutorial articles in the following topics:

- Algorithmic game theory
- Computational algebra
- Computational aspects of combinatorics and graph theory
- Computational aspects of communication
- Computational biology
- Computational complexity
- Computational geometry
- Computational learning
- Computational Models and Complexity
- Computational Number Theory
- Cryptography and information security
- Data structures
- Database theory
- Design and analysis of algorithms
- Distributed computing
- Information retrieval
- Operations Research
- Parallel algorithms
- Quantum Computation
- Randomness in Computation

Information for Librarians
Foundations and Trends® in Theoretical Computer Science, 2010, Volume 6, 4 issues. ISSN paper version 1551-305X. ISSN online version 1551-3068. Also available as a combined paper and online subscription.
Locally Decodable Codes

Sergey Yekhanin

Microsoft Research Silicon Valley, 1065 La Avenida, Mountain View, CA 94043, USA, yekhanin@microsoft.com

Abstract

Locally decodable codes are a class of “error-correcting codes.” Error-correcting codes help to ensure reliability when transmitting information over noisy channels. They allow a sender of a message to add redundancy to messages, encoding bit strings representing messages into longer bit strings called codewords, in a way that the message can still be recovered even if a certain fraction of the codeword bits are corrupted. Classical error-correcting codes however do not work well when one is working with massive messages, because their decoding time increases (at least) linearly with the length of the message. As a result in typical applications the message is first partitioned into small blocks, each of which is then encoded separately. Such encoding allows efficient random-access retrieval of the message, but yields poor noise resilience.

Locally decodable codes are codes intended to address this seeming conflict between efficient retrievability and reliability. They are codes that simultaneously provide efficient random-access retrieval and high noise resilience by allowing reliable reconstruction of an arbitrary bit of the message from looking at only a small number of randomly chosen codeword bits. This review introduces and motivates locally decodable
codes, and discusses the central results of the subject. In particular, local decodability comes at the price of certain loss in terms of code efficiency, and this review describes the currently known limits on the efficiency that is achievable.
# Contents

1 Introduction 1

1.1 Families of Locally Decodable Codes 3
1.2 Organization 5
1.3 Notes 5

2 Preliminaries 7

2.1 Locally Decodable and Locally Correctable Codes 8
2.2 Reed Muller Locally Decodable Codes 10
2.3 Summary of Parameters 16
2.4 Notes 17

3 Multiplicity Codes 19

3.1 Introduction 20
3.2 Main Results 23
3.3 The Code Construction 24
3.4 Local Correction 30
3.5 Appendix: Decoder Running Time 37
3.6 Appendix: Hasse Derivatives and Multiplicities 40
3.7 Notes 45

4 Matching Vector Codes 47

4.1 The Framework 48
4.2 Basic Decoding on Lines 50
4.3 Improved Decoding on Lines 51
4.4 Decoding on Collections of Lines 52
4.5 Binary Codes 54
4.6 Summary of Parameters 55
4.7 MV Codes versus RM Codes 62
4.8 Notes 64

5 Matching Vectors 67
5.1 Introduction 68
5.2 The Grolmusz Family 69
5.3 An Elementary Family 75
5.4 An Algebraic Family 77
5.5 Upper Bounds for Families Modulo Primes 78
5.6 Upper Bounds for Families Modulo Prime Powers 81
5.7 Upper Bounds for Families Modulo Composites 83
5.8 Notes 85

6 Lower Bounds 87
6.1 Preliminaries 87
6.2 Polynomial Lower Bound for $r$-query Codes 92
6.3 Exponential Lower Bound for 2-query Codes 94
6.4 Notes 97

7 Applications 99
7.1 Private Information Retrieval 99
7.2 Secure Multiparty Computation 105
7.3 Average-case Complexity 105
7.4 Notes 106

8 Future Directions 107
8.1 3-query Locally Decodable Codes 107
8.2 $r$-query Locally Decodable Codes 108
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.3</td>
<td>Locally Correctable Codes</td>
<td>108</td>
</tr>
<tr>
<td>8.4</td>
<td>Two Server Private Information Retrieval</td>
<td>109</td>
</tr>
<tr>
<td>8.5</td>
<td>Private Information Retrieval with Preprocessing</td>
<td>109</td>
</tr>
<tr>
<td></td>
<td><strong>Acknowledgments</strong></td>
<td>111</td>
</tr>
<tr>
<td></td>
<td><strong>References</strong></td>
<td>113</td>
</tr>
</tbody>
</table>
1

Introduction

Locally Decodable Codes (LDCs) are a special kind of error-correcting codes. Error-correcting codes are used to ensure reliable transmission of information over noisy channels as well as to ensure reliable storage of information on a medium that may be partially corrupted over time (or whose reading device is subject to errors).

In both of these applications the message is typically partitioned into small blocks and then each block is encoded separately. Such encoding strategy allows efficient random-access retrieval of the information, since one needs to decode only the portion of data one is interested in. Unfortunately, this strategy yields very poor noise resilience, since in case even a single block (out of possibly tens of thousands) is completely corrupted some information is lost. In view of this limitation, it would seem preferable to encode the whole message into a single codeword of an error-correcting code. Such solution clearly improves the robustness to noise, but is also hardly satisfactory, since one now needs to look at the whole codeword in order to recover any particular bit of the message (at least when classical error-correcting codes are used). Such decoding complexity is prohibitive for modern massive data-sets.

Locally decodable codes are error-correcting codes that avoid the problem mentioned above by having extremely efficient sublinear-time
decoding algorithms. More formally, an \( r \)-query locally decodable code \( C \) encodes \( k \)-bit messages \( x \) in such a way that one can probabilistically recover any bit \( x(i) \) of the message by querying only \( r \) bits of the (possibly corrupted) codeword \( C(x) \), where \( r \) can be as small as 2.

**Example 1.1.** The classical Hadamard code encoding \( k \)-bit messages to \( 2^k \)-bit codewords provides the simplest nontrivial example of locally decodable codes. In what follows, let \( [k] \) denote the set \( \{1,\ldots,k\} \). Every coordinate in the Hadamard code corresponds to one (of \( 2^k \)) subsets of \( [k] \) and stores the XOR of the corresponding bits of the message \( x \). Let \( y \) be an (adversarially corrupted) encoding of \( x \). Given an index \( i \in [k] \) and \( y \), the Hadamard decoder picks a set \( S \) in \( [k] \) uniformly at random and outputs the XOR of the two coordinates of \( y \) corresponding to sets \( S \) and \( S \triangle \{i\} \). (Here, \( \triangle \) denotes the symmetric difference of sets such as \( \{1,4,5\} \triangle \{4\} = \{1,5\} \), and \( \{1,4,5\} \triangle \{2\} = \{1,2,4,5\} \)). It is not difficult to verify that if \( y \) differs from the correct encoding of \( x \) in at most \( \delta \) fraction of coordinates than with probability \( 1 - 2\delta \) both decoder’s queries go to uncorrupted locations. In such case, the decoder correctly recovers the \( i \)th bit of \( x \). The Hadamard code allows for a super-fast recovery of the message bits (such as, given a codeword corrupted in 0.1 fraction of coordinates, one is able to recover any bit of the message with probability 0.8 by reading only two codeword bits).

The main parameters of interest in locally decodable codes are the codeword length and the query complexity. The length of the code measures the amount of redundancy that is introduced into the message by the encoder. The query complexity counts the number of bits that need to be read from the (corrupted) codeword in order to recover a single bit of the message. Ideally, one would like to have both of these parameters as small as possible. One however cannot minimize the length and the query complexity simultaneously. There is a trade-off. On one end of the spectrum we have classical error correcting codes that have both query complexity and codeword length proportional to the message length. On the other end we have the Hadamard code that has query complexity 2 and codeword length exponential in the message length. Establishing the optimal trade-off between the length
and the query complexity is the major goal of research in the area of locally decodable codes.

Interestingly, the natural application of locally decodable codes to data transmission and storage described above is neither the historically earliest nor the most important. LDCs have a host of applications in other areas of theoretical computer science.

### 1.1 Families of Locally Decodable Codes

One can informally classify the known families of locally decodable codes into three broad categories based on the relation between the message length \( k \) and the query complexity \( r \).

1. **Low query complexity.** Here we look at codes where \( r \) is a constant independent of \( k \) or some very slowly growing function of \( k \). Such codes have important applications in cryptography to constructions of private information retrieval schemes. Early examples of such codes are the Hadamard code and the Reed Muller (RM) code that is sketched below.

#### Reed Muller code.

The code is specified by three integer parameters, an alphabet size \( q \), a number of variables \( n \), and a degree \( d < q - 1 \). The code encodes \( k = \binom{n+d}{d} \)-long \( q \)-ary messages to \( q^n \)-long codewords. We fix a certain collection of vectors \( W = \{w_1, \ldots, w_k\} \) in \( \mathbb{F}_q^n \). A message \( x \) is encoded by a complete \( \mathbb{F}_q^n \)-evaluation of a polynomial \( F \in \mathbb{F}_q[z_1, \ldots, z_n] \) of degree up to \( d \), such that for all \( i \in [k] \), \( x(i) = F(w_i) \). Our choice of \( W \) ensures that such a polynomial exists for any \( x \). Given \( i \in [k] \) and a \( \delta \)-corrupted evaluation of \( F \) the Reed Muller decoder needs to recover the value of \( F \) at \( w_i \). To do this the decoder picks a random affine line \( L \) through \( w_i \) and reads the (corrupted) values of \( F \) at \( d+1 \) points of \( L \setminus \{w_i\} \). Next, the decoder uses univariate polynomial interpolation to recover the restriction of \( F \) to \( L \). Each query of the decoder samples a random location, thus with probability at least \( 1 - (d + 1)\delta \), it never queries a corrupted coordinate and decodes correctly. Setting \( d \) and \( q \) to be constant and letting \( n \) grow one gets \( r \)-query codes of length \( N = \exp(k^{1/(r-1)}) \).
Other families of codes in this category are the recursive codes of Beimel et al. and the Matching Vector (MV) codes. MV codes offer the best-known trade-off between the query complexity and the codeword length of locally decodable codes for small values of query complexity. In particular they give three-query codes of length $N(k)$ where $N$ grows slower than any function of the form $\exp(k^\epsilon)$. In this review we cover the construction of matching vector codes in full detail.

2. Medium query complexity. Here we look at codes with $r = \log^c k$, for some $c > 1$. Such codes have been used in constructions of probabilistically checkable proofs. They also have applications to worst-case to average-case reductions in computational complexity theory. Setting $d = n^c$, $q = \Theta(d)$ in the definition of Reed Muller codes, and letting the number of variables $n$ grow to infinity yields codes of query complexity $\log^c k$ and codeword length $N = k^{1+1/(c-1)+\alpha(1)}$. These are the best-known locally decodable codes in this regime.

3. High query complexity. Here we look at codes with $r = k^\epsilon$, for some $\epsilon > 0$. This is the only regime where we (so far) have locally decodable codes of positive rate, that is, codeword length proportional to message length. Such codes are potentially useful for data transmission and storage applications. The early examples of such codes are the Reed Muller codes with the number of variables $n = 1/\epsilon$, growing $d$, and $q = \Theta(d)$. Such setting of parameters yields codes of query complexity $r = k^\epsilon$ and rate $\epsilon^{\Theta(1/\epsilon)}$. The rate is always below $1/2$. Another family of codes in the high query complexity category is the family of multiplicity codes. Multiplicity codes are based on evaluating high degree multivariate polynomials together with their partial derivatives. Multiplicity codes extend Reed Muller codes; inherit the local-decodability of these codes, and at the same time achieve better tradeoffs and flexibility in their rate and query complexity. In particular for all $\alpha, \epsilon > 0$ they yield locally decodable codes of query complexity $r = k^\epsilon$ and rate $1 - \alpha$. In this survey we cover multiplicity codes in full detail.

Throughout the survey we use the standard notation $\exp(x) = 2^{O(x)}$.
1.2 Organization

The goal of this survey is to summarize the state-of-the-art in locally decodable codes. Our main focus is on multiplicity codes and on matching vector codes. The survey is organized into eight sections.

In Section 2 we formally define locally decodable codes and give a detailed treatment of Reed Muller codes. In Section 3 we study multiplicity codes. We show how multiplicity codes generalize Reed Muller codes and obtain bounds on their rate and query complexity.

In Section 4 we introduce the concept of matching vectors and present a transformation that turns an arbitrary family of such vectors into a family of locally decodable (matching vector) codes. We provide a detailed comparison between the parameters of matching vector codes based on the currently largest known matching families and Reed Muller codes. Section 5 contains a systematic study of families of matching vectors. We cover several constructions as well as impossibility results.

In Section 6 we deal with lower bounds for the codeword length of locally decodable codes. In Section 7 we discuss some prominent applications of locally decodable codes, namely, applications to private information retrieval schemes, secure multi party computation, and average case complexity. Finally, in the last section we list (and comment on) the most exciting open questions relating to locally decodable codes and private information retrieval schemes.

1.3 Notes

We now review the history of locally decodable codes. Ideas behind the early constructions of LDCs go back to classical codes [77, Section 10], named after their discoverers, Reed and Muller. Muller discovered the codes [70] in the 1950s, and Reed proposed the majority logic decoding [81]. Since then, local decodability of these codes has been exploited extensively. In particular, in the early 1990s a number of theoretical computer science papers developed and used local decoding algorithms for some variants of these codes [5, 11, 24, 43, 44, 67, 78]. The first formal definition of locally decodable codes was given however only in
Introduction

2000 by Katz and Trevisan [60], who cited Leonid Levin for inspiration. See also [90].

Today there are three families of locally decodable codes that surpass Reed Muller codes in terms of query complexity vs. codeword length trade-off. These are the recursive codes of Beimel et al. [15] (see also [97]), the matching vector codes [18, 19, 33, 38, 59, 61, 69, 79, 85, 99], and the multiplicity codes [63]. Matching vector codes offer the best-known trade-off between the query complexity and the codeword length of locally decodable codes for small values of query complexity. Multiplicity codes are the best-known locally decodable codes for large values of query complexity.

The first lower bounds for the codeword length of locally decodable codes were obtained in [60]. Further work on lower bounds includes [31, 41, 48, 62, 74, 94, 95, 96]. It is known that 1-query LDCs do not exist [60]. The length of optimal 2-query LDCs was settled in [62] and is exponential in the message length. However, for values of query complexity \( r \geq 3 \) we are still very far from closing the gap between lower and upper bounds. Specifically, the best lower bounds to date are of the form \( \tilde{\Omega}(k^{1+1/(\lceil r/2 \rceil - 1)}) \) due to [95], while the best upper bounds are super-polynomial in \( k \) when \( r \) is a constant [38, 69].
References


References


References


References


References


References


