Relaxed Locally Correctable Codes in Computationally Bounded Channels*

Jeremiah Blocki†, Venkata Gandikota‡, Elena Grigorescu†, and Samson Zhou§
†Department of Computer Science, Purdue University. {elena-g, jblocki}@purdue.edu
‡College of Information and Computer Science, University of Massachusetts. gandikota.venkata@gmail.com
§School of Informatics, Computing, and Engineering, Indiana University. samsonzhou@gmail.com

***Abstract*—Error-correcting codes that admit *local* decoding and correcting algorithms have been the focus of much recent research due to their numerous applications. An important goal is to obtain the best possible tradeoffs between the number of symbols of the codeword that the local decoding algorithm must examine (the *locality* of the task), and the amount of redundancy in the encoding (the *information rate*).**

In Hamming's classical adversarial channel model, the current tradeoffs are dramatic, allowing either small locality, but superpolynomial blocklength, or small blocklength, but high locality. However, in the computationally bounded, adversarial channel model, proposed by Lipton (STACS 1994), constructions of locally decodable codes suddenly exhibit small locality and small blocklength, but these constructions require strong trusted setup assumptions e.g., Ostrovsky, Pandey and Sahai (ICALP 2007) construct private locally decodable codes in the setting where the sender and receiver already share a symmetric key.

We study variants of locally decodable and locally correctable codes in computationally bounded, adversarial channels, in a setting with no public-key or private-key cryptographic setup. The only setup assumption we require is the selection of the *public* parameters (seed) for a collision-resistant hash function. Specifically, we provide constructions of *relaxed locally correctable* and *relaxed locally decodable codes* over the binary alphabet, with constant information rate, and poly-logarithmic locality.

Our constructions, which compare favorably with their classical analogs, crucially employ *collision-resistant hash functions* and *local expander graphs*, extending ideas from recent cryptographic constructions of memory-hard functions.

I. Introduction

Classically, an error-correcting code is a tuple $(\mathsf{Enc}, \mathsf{Dec})$ of encoding and decoding algorithms employed by a sender to encode messages, and by a receiver to decode them, after potential corruption by a noisy channel during transmission. Specifically, the sender encodes a *message* m of k symbols from an alphabet Σ into a *codeword* c of block-length n consisting of symbols over the same alphabet, via $\mathsf{Enc} : \Sigma^k \to \Sigma^n$. The receiver uses $\mathsf{Dec} : \Sigma^n \to \Sigma^k$ to recover the message from a received word $w \in \Sigma^n$, a corrupted version of some $\mathsf{Enc}(m)$. Codes over the binary alphabet $\Sigma = \{0, 1\}$ are preferred in practice. The quantities of interest in designing classical codes are the *information rate*, defined as k/n, and the *error rate*, which is the tolerable fraction of errors in the received word. Codes with both large information rate and large error rate are most desirable.

In modern uses of error-correcting codes, one may only need to recover small portions of the message, such as a single bit. In such settings, the decoder may not need to read the entire received word $w \in \Sigma^n$, but only read a few bits of it. Given an index $i \in [n]$ and oracle access to w, a local decoder must make only $q = o(n)$ queries into w, and output the bit m_i. Codes that admit such fast decoders are called *locally decodable codes* ($\mathsf{LDC}s$) [1], [2]. The parameter q is called the *locality* of the decoder. A related notion is that of *locally correctable codes* ($\mathsf{LCC}s$). $\mathsf{LCC}s$ are codes for which the local decoder with oracle access to w must output bits of the codeword c, instead of bits of the message m. $\mathsf{LDC}s$ and $\mathsf{LCC}s$ have widespread applications in many areas of theoretical computer science, including private information retrieval, probabilistically checkable proofs, self-correction, fault-tolerant circuits, hardness amplification, and data structures (e.g., [3]–[9] and surveys [10], [11]). However, constructions of such codes suffer from apparently irreconcilable tension between locality and rate: existing codes with constant locality have slightly subexponential blocklength [12]–[14], and

*A full version of this paper is accessible at: https://arxiv.org/abs/1803.05652

This work was done in part while V.G. was a postdoc at the Department of Computer Science, JHU; and S.Z. was a graduate student at the Department of Computer Science, Purdue University. Research of E.G. and S.Z. was supported in part by NSF CCF-1649515 and by a grant from the Purdue Research Foundation. Research of J.B was supported in part by NSF CNS-1755708.

codes of linear blocklength have slightly subpolynomial query complexity [15]. For example, see surveys by Yekhanin [16] and by Kopparty and Saraf [17].

Ben-Sasson *et al.* [18] propose the notion of *relaxed locally decodable codes* (RLDCs) that remedies the dramatic tradeoffs of classical LDCs. In this notion the decoding algorithm is allowed to output $\perp$ sometimes, to signal that it does not know the correct value; however, it should not output an incorrect value too often. More formally, given $i \in [k]$ and oracle access to the received word w assumed to be relatively close to some codeword $c = \mathsf{Enc}(m) \in \Sigma^n$, the local decoder (1) outputs m_i if $w = c$; (2) otherwise, with probability $2/3$ outputs either m_i or $\perp$; and (3) the set of indices i such that the decoder outputs m_i (the correct value) with probability $2/3$ is of size $> \rho \cdot k$ for some constant $\rho > 0$. The relaxed definition allows them to achieve RLDCs with constant query complexity and blocklength $n = k^{1+\epsilon}$.

Recently, Gur *et al.* [19] introduce the analogous notion of *relaxed locally correctable codes* (RLCCs). In particular, upon receiving a word $w \in \Sigma^n$ assumed to be close to some codeword c, the decoder: (1) outputs c_i if $w = c$; (2) outputs either c_i or $\perp$ with probability $2/3$, otherwise; and (3) the set of indices i such that the decoder outputs c_i with probability $2/3$ is of size $\rho \cdot n$, for some $\rho > 0$. In fact, [19] omits condition (3) in their definition, since the first two conditions imply the 3rd, for codes with constant locality that can withstand a constant fraction of error [18]. The reduction from [18], however, does not maintain the asymptotic error rate, and in particular, in the non-constant query complexity regime, the error rate becomes subconstant. Since our results work in the $\omega(1)$-query regime, we will build codes that achieve the 3rd condition as well (for constant error rate). The results in [19] obtain significantly better parameters for RLCCs than for classical LCCs; namely, they construct RLCCs with constant query complexity, polynomial block length, and constant error rate, and RLCCs with quasipolylogarithmic query complexity, linear blocklength (constant rate), with the caveat that the error rate is subconstant. These results immediately extend to RLDCs, since their codes are *systematic*, meaning that the initial part of the encoding consists of the message itself.

In this work we study RLDCs and RLCCs in the more restricted, yet natural, *computationally bounded adversarial channel*, introduced by Lipton [20]. All the above constructions of local codes assume a channel that may introduce a bounded number of adversarial errors, and the channel has as much time as it needs to decide what positions to corrupt. Lipton argued that one can always reasonably assume that an adversarial channel is computationally bounded and can be modeled as *polynomial time probabilistic* (PPT) algorithms. Variants of this model have been initially studied for classical error-correcting codes [20]–[24] to show better error rate capabilities than in the Hamming model. See full version for further details on related work [25].

Other work has focused on the construction of locally decodable codes when the sender and receiver have already exchanged cryptographic keys. Ostrovsky *et al.* [26] construct a "private key" locally decodable codes with constant rate and a small (superconstant) locality against computationally bounded channels. Similarly, Hemenway and Ostrovky [27] and Hemenway *et al.* [28] construct public-key LDCs i.e., the encoding algorithm uses a secret key, but the decoder only needs to know the sender's public key. A crucial difference is that in our constructions of RLDCs and RLCCs, the encoding algorithm is completely public and *does not* require the sender to use any secret key. Instead our constructions only rely on the existence of a collision-resilient hash function H, which will be used in both the encoding and decoding algorithms. In practice, one could instantiate H with SHA3 and no further setup assumptions are needed.

II. Our Contributions

We introduce the notions of a *computationally relaxed locally correctable codes* (CRLCCs) which is analogous to RLCCs when the channel is computationally bounded. We construct CRLCCs with constant information and error rates as well as polylogarithmic locality, improving on [19] for bounded channels. Our codes are systematic, and therefore give *computationally relaxed locally decodable codes* (CRLDCs).

Since these codes interact with an adversarial channel, their strength is not only measured in their error correction and locality capabilities (as is the case for RLCCs/RLDCs in the Hamming channel), but also in the security they provide against the channel. We present these codes while describing how they interact with the channel, in order to make the analogy with the classical setting. We use the notation Enc and Dec to denote encoding and decoding algorithms.

Definition 2.1: A *local code* is a tuple $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ of probabilistic algorithms such that

- $\mathsf{Gen}(1^\lambda)$ takes as input security parameter λ and generates a public seed $s \in \{0,1\}^*$. This public seed s is *fixed* once and for all.

- Enc takes as input the public seed s and a message $x \in \Sigma^k$ and outputs a codeword $c = \mathsf{Enc}(s, x)$ with $c \in \Sigma^n$.
- Dec takes as input the public seed s, an index $i \in [n]$, and is given oracle access to a word $w \in \Sigma^n$. $\mathsf{Dec}^w(s, i)$ outputs a symbol $b \in \Sigma$ (which is supposed to be the value at position i of the closest codeword to w).

We say that the code is *efficient* if $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ are all probabilistic polynomial time (PPT) algorithms, and we say that the (information) *rate* of the code is k/n.

Definition 2.2: A *computational adversarial channel* $\mathcal{A}$ with error rate τ is an algorithm that interacts with a local code $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ in rounds, as follows. In each round of the execution, given a security parameter λ,

(1) Generate $s \leftarrow \mathsf{Gen}(1^\lambda)$; s is public, so Enc, Dec, and $\mathcal{A}$ have access to s. This public seed s is *fixed* once and for all.
(2) The channel $\mathcal{A}$ on input s hands a message x to the sender.
(3) The sender computes $c = \mathsf{Enc}(s, x)$ and hands it back to the channel (in fact the channel can compute c without this interaction).
(4) The channel $\mathcal{A}$ corrupts at most τn entries of c to obtain a word $w \in \Sigma^n$ and selects a challenge index $i \in [n]$; w is given to the receiver's Dec with query access along with the challenge index i.
(5) The receiver outputs $b \leftarrow \mathsf{Dec}^w(s, i)$.
(6) We define $\mathcal{A}(s)$'s *probability of fooling* Dec on this round to be $p_{\mathcal{A},s} = \Pr[b \notin \{\perp, c_i\}]$, where the probability is taken only over the randomness of the $\mathsf{Dec}^w(s, i)$. We say that $\mathcal{A}(s)$ is γ-successful *at fooling* Dec if $p_{\mathcal{A},s} > \gamma$. We say that $\mathcal{A}(s)$ is ρ-successful *at limiting* Dec if $|\mathsf{Good}_{\mathcal{A},s}| < \rho \cdot n$, where $\mathsf{Good}_{\mathcal{A},s} \subseteq [n]$ is the set of indices j such that $\Pr[\mathsf{Dec}^w(s, j) = c_j] > \frac{2}{3}$. We use $\mathsf{Fool}_{\mathcal{A},s}(\gamma, \tau, \lambda)$ (resp. $\mathsf{Limit}_{\mathcal{A},s}(\rho, \tau, \lambda)$) to denote the event that the attacker was γ-successful at fooling Dec (resp. ρ-successful at limiting Dec) on this round.

We now define our secure RLCC codes against computational adversarial channels.

Definition 2.3 ((Computational) Relaxed Locally Correctable Codes (CRLCC)): A local code $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a $(q, \tau, \rho, \gamma(\cdot), \mu(\cdot))$-CRLCC *against a class* $\mathbb{A}$ of adversaries if Dec^w makes at most q queries to w and satisfies the following:

(1) For all public seeds s if $w \leftarrow \mathsf{Enc}(s, x)$ then $\mathsf{Dec}^w(s, i)$ outputs $b = (\mathsf{Enc}(s, x))_i$.
(2) For all $\mathcal{A} \in \mathbb{A}$ we have $\Pr[\mathsf{Fool}_{\mathcal{A},s}(\gamma(\lambda), \tau, \lambda)] \leq \mu(\lambda)$, where the randomness is taken over the selection of $s \leftarrow \mathsf{Gen}(1^\lambda)$ as well as $\mathcal{A}$'s random coins.
(3) For all $\mathcal{A} \in \mathbb{A}$ we have $\Pr[\mathsf{Limit}_{\mathcal{A},s}(\rho, \tau, \lambda)] \leq \mu(\lambda)$, where the randomness is taken over the selection of $s \leftarrow \mathsf{Gen}(1^\lambda)$ as well as $\mathcal{A}$'s random coins.

When $\mu(\lambda) = 0$, $\gamma(\lambda) = \frac{1}{3}$ is a constant and $\mathbb{A}$ is the set of all (computationally unbounded) channels, we say that the code is a (q, τ, ρ, γ)-RLCC. When $\mu(\cdot)$ is a negligible function *and* $\mathbb{A}$ is restricted to the set of all probabilistic polynomial time (PPT) attackers, we say that the code is a (q, τ, ρ, γ)-CRLCC (computational relaxed locally correctable code).

We say that a code that satisfies conditions 1 and 2 is a *Weak CRLCC*, while a code that satisfies conditions 1, 2 and 3 is a *Strong CRLCC* code.

We construct Weak and Strong CRLCCs against PPT adversaries, under the assumption that *Collision-Resistant Hash Functions* (CRHF) exist. Briefly, a CRHF function is a pair (GenH, H) of probabilistic polynomial time (PPT) algorithms, where GenH takes as input a security parameter λ and outputs a public seed $s \in \{0,1\}^*$; the function $H : \{0,1\}^* \times \Sigma^* \to \Sigma^{\ell(\lambda)}$, takes as input the seed s and a long enough input that is hashed into a string of length $\ell(\lambda)$. We note that H is deterministic upon fixing s. The value $\ell(\lambda)$ is the *length* of the hash function. (GenH, H) is said to be collision-resistant if for all PPT adversaries that take as input the seed s generated by $\mathsf{Gen}(1^\lambda)$, the probability that a collision pair (x, x') is produced, i.e. such that $H(s, x) = H(s, x')$ and $x \neq x'$, is negligible in 1^λ.

Theorem 2.4, our main result, states that it is possible to construct a constant rate Strong CRLDC with polylog locality that withstands a constant error rate τ. Furthermore, the information rate $r(\tau)$ approaches 1 as τ approaches 0. By contrast, the classical RLCCs of [19] achieve constant information rate, but subconstant error rate and $(\log n)^{\mathcal{O}(\log \log n)}$ query complexity in the Hamming channel.

Theorem 2.4: Assuming the existence of a collision-resistant hash function (GenH, H) with length $\ell(\lambda)$, there exists a constant $0 < \tau' < 1$ and negligible functions $\mu(\cdot), \gamma(\cdot)$ such that for all $\tau \leq \tau'$ there exist constants $0 < r(\tau), \rho(\tau) < 1$ such that there exists a $(\ell(\lambda) \cdot \mathsf{polylog}\, n, \tau, \rho(\tau), \gamma(\cdot), \mu(\cdot))$-Strong CRLCC of blocklength n over the binary alphabet with rate $r(\tau)$ where $r(\cdot)$ and $\rho(\cdot)$ have the property that $\lim_{\tau \to 0} r(\tau) = 1$ and $\lim_{\tau \to 0} \rho(\tau) = 1$. In particular,

if $\ell(\lambda) = \mathsf{polylog}\,\lambda$ and $\lambda \in \Theta(n)$ then the code is a $(\mathsf{polylog}\,n, \tau, \rho(\tau), \gamma(\cdot), \mu(\cdot))$-Strong CRLCC.

Our constructions are systematic and imply the existence of a Strong CRLDC with the same parameters.

III. Technical Ingredients

At a technical level our construction uses two main building blocks: *local expander graphs* and *collision resistant hash functions*. Below we describe how we use these tools to construct Weak CRLCCs. We will also briefly sketch how the construction can be modified to obtain Strong CRLCCs though we defer a complete treatment to the full version of the paper [25].

Weak CRLCCs (Encoding): Our construction of Weak CRLCCs involves labeling a k-node directed acyclic graph G with a collision resistant hash function $H(s,\cdot)$. We require that G satisfies key combinatorial property called δ-local expansion described later. Given an input word $x = (x_1 \circ \ldots \circ x_k)$ (broken up into bit strings of length $\ell(\lambda)$) and a k node local expander graph G, the label of node v with parents $v_1, \ldots, v_d$ is computed as $\ell_{v,s} = H\left(s, x_v \circ \ell_{v_1,s} \circ \ldots \circ \ell_{v_d,s}\right) \in \{0,1\}^{\ell(\lambda)}$, where $\ell_{v_1,s}, \ldots, \ell_{v_d,s}$ are the labels of the parents of node v (if any exist), and $\circ$ denotes string concatenation. When $\ell(\lambda) \in \mathcal{O}(\mathsf{polylog}\,\lambda)$ we will select $\lambda \in \mathcal{O}(n)$ to ensure that $\ell(\lambda) \in \mathcal{O}(\mathsf{polylog}\,n)$. We use the notation Enc and Dec for the encoding and decoding of our construction, while we use ECC and ECCD to denote the efficient encoding and decoding algorithms for a good binary code with constant rate and relative distance (e.g., [29], [30]), which can decode efficiently from some constant fractions of errors.

We first apply ECC to $x_1, \ldots, x_k$ to obtain codewords $c_1, \ldots, c_k \in \{0,1\}^{\mathcal{O}(\ell(\lambda))}$ where $c_i = \mathsf{ECC}(x_i)$. Also, for $v \in [k]$ we let $c_{v+k} = \mathsf{ECC}(\ell_{v,s})$ which is the encoding of the label corresponding to the node v in G. The final output is $c = (c_1 \circ \ldots \circ c_{2k-1} \circ c_{2k} \circ c_{2k+1} \circ \ldots \circ c_{3k})$ where $c_{2k+1} = \ldots = c_{3k} = c_{2k}$ consists of k copies of the last codeword c_{2k}. The final word is an n bit message with $n = \mathcal{O}(k\ell(\lambda))$. By repeating this last codeword k times we ensure that it is *not possible* for the attacker to irreparably corrupt the final label $\ell_{k,s}$.

Weak CRLCCs (Local Decoder): Given a (possibly corrupted) codeword c' produced by a PPT attacker $\mathcal{A}$ we let $x' = (x'_1 \circ \ldots \circ x'_k)$ with $x'_i = \mathsf{ECCD}(c'_i)$ (possibly $\perp$) and we let $\ell'_{v,s} = \mathsf{ECCD}(c'_{v+k})$ for $v \in [k]$ and $\ell'_{k,s,j} = \mathsf{ECCD}(c'_{2k+j})$ for each $j \in [k]$. We say that a node v is *green* if it is locally consistent i.e., $\ell'_{v,s} = H\left(s, x'_v \circ \ell'_{v_1,s} \circ \ldots \circ \ell'_{v_d,s}\right)$, otherwise, we say that the node is *red*. Because $H(s,\cdot)$ is collision resistant it is not too difficult to prove that if a green node has the *correct* label $\ell'_{v,s} = \ell_{v,s}$ then it *must* be the case that $x'_v = x_v$ and $\ell'_{v_i,s} = \ell_{v_i,s}$ for each of v's parents $v_1, \ldots v_d$ — otherwise $\mathcal{A}$ would have found a hash collision! If a graph contains too many red nodes then this is easily detectable by random sampling and our *weak* local decoder is allowed to output $\perp$ if it detects *any* red nodes.

Our local decoder will first obtain the final label $\ell_{k,s}$ by random sampling some labels from $\ell'_{k,s,1}, \ldots, \ell'_{k,s,k}$ and checking to make sure each of these labels is equal to $\ell'_{k,s}$. If this check passes then with high probability we must have $\ell'_{k,s} = \ell_{k,s}$ since the attacker cannot corrupt too many of these labels. The local decoder also checks that the node k is green i.e., $\ell'_{k,s}$ is *locally consistent*. If $\ell'_{k,s}$ is *correct* and node k is *green* then this node serves as an anchor point. In particular, if node v and node k are connected by a green path then we can conclude that this part of the codeword has not been corrupted i.e., $\ell_{v,s} = \ell_{v,s}$ and that $x_v = x'_v$. Thus, the remaining challenge is to design a test with small locality that outputs 0 (whp) whenever there is no green path connecting v to k and *always* outputs 1 when there are no red nodes. We show how to design such a test whenever G is a local expander.

Local Expander Graphs: Given a graph $G = (V, E)$ and distinguished subsets $A, B \subseteq V$ of nodes such that $A \cap B = \emptyset$ and $|A| \leq |B|$ we say that the pair (A, B) contains a δ-*expander* if for all $X \subseteq A$ and $Y \subseteq B$ with $|X| > \delta|B|$ and $|Y| > \delta|B|$ there is an edge connecting X and Y. We say that a DAG G is a δ-*local expander* around a node v if for *any* radius $r > 0$ and *any* node $v \geq 2r$, the contiguous pair $A = \{v-2r+1, \ldots, v-r\}$ and $B = \{v-r+1, \ldots, v\}$ contain a δ-expander *and* the contiguous pair $C = \{v, \ldots, v+r-1\}$ and $D = \{v+r, \ldots, v+2r-1\}$ contain a δ-expander. When this property holds for *every* node $v \in V(G) = \{1, \ldots, n\}$ we simply say that the DAG G is a δ-local expander. For any constant $\delta > 0$ it is possible to (explicitly) construct a δ-local expander with the property that $\mathsf{indeg}(G), \mathsf{outdeg}(G) \in \mathcal{O}(\log n)$ [31], [32]. Local expanders have several nice properties that have been recently exploited in cryptography in the design and analysis of data-independent memory hard functions (iMHFs) [32]–[36], proofs of sequential work [37] and proofs of space [38].

If G is a δ-local expander (for a suitably small constant $\delta > 0$) and we delete a *large* number of nodes $S \subseteq V$, then we can still prove that (1) at least $n - \frac{2|S|}{\alpha}$ of the nodes have the property that they are "α-good" with respect to the deleted set S, and (2) *any* pair of α-good

nodes u and v are connected by a directed path. Here, a node v is α-good with respect to S if, for any radius $r < v$, we have at most αr nodes in $S \cap [v - r + 1, v]$ *and* for any radius $r \leq n - v + 1$, we have at most αr nodes in $S \cap [v, v + r - 1]$.

Recall that given a (potentially corrupted) codeword, we defined a node v to be green if the hash label was locally consistent. Now if we let S denote the set of red nodes then we can develop an *efficient* randomized testing algorithm to check if a particular node v is α-good or not. (We defer to the full version [25] the details of this test and of the full construction of Strong CRLCCs using the ideas developed above.)

REFERENCES

[1] J. Katz and L. Trevisan, "On the efficiency of local decoding procedures for error-correcting codes," in *STOC*, 2000, pp. 80–86. I

[2] M. Sudan, L. Trevisan, and S. P. Vadhan, "Pseudorandom generators without the XOR lemma (abstract)," in *CCC*, 1999, p. 4. I

[3] L. Babai, L. Fortnow, L. A. Levin, and M. Szegedy, "Checking computations in polylogarithmic time," in *STOC*, 1991, pp. 21–31. I

[4] C. Lund, L. Fortnow, H. J. Karloff, and N. Nisan, "Algebraic methods for interactive proof systems," *J. ACM*, vol. 39, no. 4, pp. 859–868, 1992. I

[5] M. Blum, M. Luby, and R. Rubinfeld, "Self-testing/correcting with applications to numerical problems," *J. Comput. Syst. Sci.*, vol. 47, no. 3, pp. 549–595, 1993. I

[6] M. Blum and S. Kannan, "Designing programs that check their work," *J. ACM*, vol. 42, no. 1, pp. 269–291, 1995. I

[7] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *J. ACM*, vol. 45, no. 6, pp. 965–981, 1998. I

[8] V. Chen, E. Grigorescu, and R. de Wolf, "Error-correcting data structures," *SIAM J. Comput.*, vol. 42, no. 1, pp. 84–111, 2013. I

[9] A. Andoni, T. Laarhoven, I. P. Razenshteyn, and E. Waingarten, "Optimal hashing-based time-space trade-offs for approximate near neighbors," in *SODA*, 2017, pp. 47–66. I

[10] L. Trevisan, "Some applications of coding theory in computational complexity," *CoRR*, vol. cs.CC/0409044, 2004. I

[11] W. I. Gasarch, "A survey on private information retrieval (column: Computational complexity)," *Bulletin of the EATCS*, vol. 82, pp. 72–107, 2004. I

[12] S. Yekhanin, "Towards 3-query locally decodable codes of subexponential length," *J. ACM*, vol. 55, no. 1, pp. 1:1–1:16, 2008. I

[13] K. Efremenko, "3-query locally decodable codes of subexponential length," *SIAM J. Comput.*, vol. 41, no. 6, pp. 1694–1703, 2012. I

[14] Z. Dvir, P. Gopalan, and S. Yekhanin, "Matching vector codes," *SIAM J. Comput.*, vol. 40, no. 4, pp. 1154–1178, 2011. I

[15] S. Kopparty, O. Meir, N. Ron-Zewi, and S. Saraf, "High-rate locally correctable and locally testable codes with sub-polynomial query complexity," *J. ACM*, vol. 64, no. 2, pp. 11:1–11:42, 2017. I

[16] S. Yekhanin, "Locally decodable codes," *Foundations and Trends in Theoretical Computer Science*, vol. 6, no. 3, pp. 139–255, 2012. I

[17] S. Kopparty and S. Saraf, "Guest column: Local testing and decoding of high-rate error-correcting codes," *SIGACT News*, vol. 47, no. 3, pp. 46–66, 2016. I

[18] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. P. Vadhan, "Robust pcps of proximity, shorter pcps, and applications to coding," *SIAM J. Comput.*, vol. 36, no. 4, pp. 889–974, 2006, a preliminary version appeared in the Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC). I

[19] T. Gur, G. Ramnarayan, and R. D. Rothblum, "Relaxed locally correctable codes," in *ITCS*, 2018, pp. 27:1–27:11. I, II, II

[20] R. J. Lipton, "A new approach to information theory," in *STACS*, 1994, pp. 699–708. I

[21] Y. Ding, P. Gopalan, and R. Lipton, "Error correction against computationally bounded adversaries," 2004, manuscript. I

[22] S. Micali, C. Peikert, M. Sudan, and D. A. Wilson, "Optimal error correction against computationally bounded noise," in *TCC*, 2005, pp. 1–16. I

[23] V. Guruswami and A. D. Smith, "Optimal rate code constructions for computationally simple channels," *J. ACM*, vol. 63, no. 4, pp. 35:1–35:37, 2016. I

[24] R. Shaltiel and J. Silbak, "Explicit list-decodable codes with optimal rate for computationally bounded channels," in *APPROX/RANDOM*, 2016, pp. 45:1–45:38. I

[25] J. Blocki, V. Gandikota, E. Grigorescu, and S. Zhou, "Relaxed locally correctable codes in computationally bounded channels," *CoRR*, vol. abs/1803.05652, 2018. I, III, III

[26] R. Ostrovsky, O. Pandey, and A. Sahai, "Private locally decodable codes," in *ICALP*, 2007, pp. 387–398. I

[27] B. Hemenway and R. Ostrovsky, "Public-key locally-decodable codes," in *CRYPTO*, 2008, pp. 126–143. I

[28] B. Hemenway, R. Ostrovsky, M. J. Strauss, and M. Wootters, "Public key locally decodable codes with short keys," in *RANDOM*, 2011, pp. 605–615. I

[29] J. Justesen, "Class of constructive asymptotically good algebraic codes," *IEEE Trans. Information Theory*, vol. 18, no. 5, pp. 652–656, 1972. III

[30] M. Sipser and D. A. Spielman, "Expander codes," *IEEE Trans. Information Theory*, vol. 42, no. 6, pp. 1710–1722, 1996. III

[31] P. Erdös, R. L. Graham, and E. Szemeredi, "On sparse graphs with dense long paths." Stanford University, Stanford, CA, USA, Tech. Rep., 1975. III

[32] J. Alwen, J. Blocki, and K. Pietrzak, "Sustained space complexity," in *EUROCRYPT 2018, Part II*, ser. LNCS, J. B. Nielsen and V. Rijmen, Eds., vol. 10821. Springer, Heidelberg, Apr. / May 2018, pp. 99–130. III

[33] J. Alwen, J. Blocki, and B. Harsha, "Practical graphs for optimal side-channel resistant memory-hard functions," in *ACM CCS 17*, B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds. ACM Press, Oct. / Nov. 2017, pp. 1001–1017. III

[34] J. Alwen, J. Blocki, and K. Pietrzak, "Depth-robust graphs and their cumulative memory complexity," in *EUROCRYPT 2017, Part III*, ser. LNCS, J. Coron and J. B. Nielsen, Eds., vol. 10212. Springer, Heidelberg, Apr. / May 2017, pp. 3–32. III

[35] J. Blocki and S. Zhou, "On the depth-robustness and cumulative pebbling cost of Argon2i," in *TCC 2017, Part I*, ser. LNCS, Y. Kalai and L. Reyzin, Eds., vol. 10677. Springer, Heidelberg, Nov. 2017, pp. 445–465. III

[36] J. Alwen and J. Blocki, "Efficiently computing data-independent memory-hard functions," in *CRYPTO 2016, Part II*, ser. LNCS, M. Robshaw and J. Katz, Eds., vol. 9815. Springer, Heidelberg, Aug. 2016, pp. 241–271. III

[37] M. Mahmoody, T. Moran, and S. P. Vadhan, "Publicly verifiable proofs of sequential work," in *ITCS 2013*, R. D. Kleinberg, Ed. ACM, Jan. 2013, pp. 373–388. III

[38] S. Dziembowski, S. Faust, V. Kolmogorov, and K. Pietrzak, "Proofs of space," in *CRYPTO 2015, Part II*, ser. LNCS, R. Gennaro and M. J. B. Robshaw, Eds., vol. 9216. Springer, Heidelberg, Aug. 2015, pp. 585–605. III