Private Locally Decodable Codes*

Rafail Ostrovsky Omkant Pandey Amit Sahai

Department of Computer Science
University of California, Los Angeles 90095

{rafail,omkant,sahai}@cs.ucla.edu

Abstract

We consider the problem of constructing efficient locally decodable codes in the presence of a computationally bounded adversary. Assuming the existence of one-way functions, we construct *efficient* locally decodable codes with positive information rate and *low* (almost optimal) query complexity which can correctly decode any given bit of the message from constant channel error rate ρ. This compares favorably to our state of knowledge locally-decodable codes without cryptographic assumptions. For all our constructions, the probability for any polynomial-time adversary, that the decoding algorithm incorrectly decodes any bit of the message is negligible in the security parameter.

1 Introduction

When a *message* $\mathbf{x}$ is sent over a *channel* $\mathcal{C}$, the channel might introduce some *errors* so that the received message differs from the original message $\mathbf{x}$. To deal with this, the *sender* typically encodes the given message to obtain a *codeword* $\mathbf{y}$ so that $\mathbf{x}$ can be recovered even if the received codeword $\mathbf{y}'$ differs from the original encoding $\mathbf{y}$ in some of the places.

The message is represented by a sequence of k symbols from alphabet Σ. The codeword is also represented as a sequence of K symbols from the same[1] alphabet Σ. The encoding function is denoted by $\mathcal{S} : \Sigma^k \to \Sigma^K$ and the decoding function is denoted by $\mathcal{R} : \Sigma^K \to \Sigma^k$. The *information rate* (or simply *rate*) of the code is k/K and measures the amount of extra information needed by the code for correctly decoding from errors. Such a coding scheme is called a $(K, k)_q$-coding scheme, where $q = |\Sigma|$.

When the whole message $\mathbf{x}$ should be recovered from the corrupted codeword $\mathbf{y}'$, the decoding algorithm reads $\mathbf{y}'$ entirely. If one is interested in reading only one bit of $\mathbf{x}$, more efficient coding schemes are possible. In particular, it is possible to construct codes which can decode a single bit of $\mathbf{x}$ by reading only a few bits of $\mathbf{y}'$. Such codes are called locally decodable codes (LDCs) [1, 24, 13].

Informally, a locally decodable code with *query complexity* ℓ, *error rate* ρ, and error correction probability p is a pair of algorithms $(\mathcal{S}, \mathcal{R})$, where $\mathcal{S}$ is the encoding algorithm and $\mathcal{R}$ is the decoding algorithm, such that the decoding algorithm makes at most ℓ queries into the corrupted codeword $\mathbf{y}'$ and recovers any given bit j of $\mathbf{x}$ with probability p or more if $\mathbf{y}'$ differs from $\mathbf{y}$ in at most a ρ fraction of alphabets. For brevity, such a code is sometimes written as (ℓ, ρ, p)-LDC and we require

*Part of this work was done when all the authors were at IPAM. The first author is supported in part by NSF Cybertrust grant No. 0430254, Xerox Innovation group Award and IBM Faculty Award. The second and third authors were supported in part from grants from the NSF ITR and Cybertrust programs (including grants 0627781, 0456717, and 0205594), a subgrant from SRI as part of the Army Cyber-TA program, an equipment grant from Intel, and an Alfred P. Sloan Foundation Research Fellowship.

[1]For simplicity we assume that the two alphabets are the same, although they can be different.

that $p > 0.5$. An LDC is called *adaptive* if the queries of $\mathcal{R}$ depend upon the answers of previous queries. It is called *non-adaptive* if they depend only on the random coins of $\mathcal{R}$.

Of course, locally decodable codes with high information rate, high error rate, high error correction probability, and low query complexity are most desirable. Low alphabet sizes ($q = |\Sigma|$) are desirable too as most channels are best at transmitting only bits.

Locally decodable codes have found several notable applications. In complexity theory they have been useful in self-correcting computations [7, 8], probabilistically checkable proof systems [1], worst-case to average-case hardness reductions in the constructions of pseudo-random generators [2, 25], and so on. In cryptography they have been useful due to their interesting connection with private information retrieval protocols [5, 15, 18, 3]. Their interesting properties make them applicable in several database applications such as fault-tolerant data storage [13]. It is tempting to say that constructions of good locally decodable codes can yield benefits to several related fields of computer science.

Modeling the Noisy Channel. The nature of channel errors plays an important role in the design of good error correcting codes. Historically, there are two popular ways of modeling a noisy channel: Shannon's model and Hamming's model. In Shannon's *symmetric channel* model, each symbol is changed to a random different one independently with some fixed probability. In Hamming's *adversarial channel* model, symbols get changed in the worst possible manner subject to an upper bound on the number of errors (such as a constant fraction of the size of the codeword). It should be noted that Hamming's channel are computationally *unbounded*. As a consequence, good error-correcting codes in Hamming's model ensure robustness of the coding scheme. But at the same time, constructing error correcting codes becomes more challenging in this model. In particular, *good*[2] locally decodable codes are not known to exist in this model.

An interesting idea due to Lipton [17], models the noisy channel as a computationally *bounded* adversarial channel. That is, the channel $\mathcal{C}$ is modeled as a *probabilistic polynomial time* algorithm which can change symbols in the worst possible manner subject to an upper bound on the number of errors. Thus, Lipton's channels are essentially Hamming channels restricted to feasible computation. Modeling channels in this way makes a lot of sense as *all* real world channels are actually computationally bounded. The codes designed in this model guarantee that if a channel can cause incorrect decoding with high probability, it can also be used to break standard hardness assumptions.

Working with such computationally bounded channels has led to several interesting results of late. In particular, Gopalan, Lipton, and Ding [11] develop a technique called *code scrambling* which recovers from high error rates by using few shared random bits. Similarly, Micali, Peikert, Sudan, and Wilson construct codes that can uniquely decode from error-rates beyond the classical bounds. Other notable results that use the idea of shared randomness in particular, are the results of Langberg [16] and Smith [23]. We remark that all these results are for standard error correcting codes and *not* for locally decodable codes. In this paper we continue in this important direction and construct good LDCs against computationally bounded noisy channels. We shall summarize our results shortly.

Previous Work On Locally Decodable Codes. In order to understand the significance of our results, it is important that we shed some light on previous work related to locally decodable codes. Mainly, there has been two important research directions in this area: proving lower bounds on the size of LDCs and constructing good LDCs. All prior work in this area deals with computationally unbounded channels.

The first direction investigates the relation between the code length K and the message length k for (ℓ, ρ, p)-LDCs. Katz and Trevisan [13] first started investigating this direction and showed that

[2]By good LDCs we mean LDCs with high information rate and high probability of error correction with small query size and constant error rate.)

for non-adaptive LDCs, K is at least $k^{1+\frac{1}{\ell-1}}$ (suppressing the dependence on ρ and p). Deshpande et al [6] showed that this bound holds even for adaptive LDCs. Currently, the best known lower bounds for general locally decodable codes are due to Woodruff [27] who shows that $K = \Omega\left(\frac{k^{1+\frac{2}{\ell-1}}}{\log k}\right)$. A series of papers [10, 20, 21, 14, 26] concentrated on LDCs with $\ell = 2$ (or 3) and established exponential lower bounds. In particular for 2-query LDCs $K = \exp(\Omega(\frac{\rho}{2-2p}k))$.

The other direction focussed on *constructing* the locally decodable codes. Important constructions in this direction for *constant* query length appeared in [3, 4]. Unfortunately, all these constructions yield codes that are exponentially long in k_i. Currently, the best known construction is due to Yekhanin [28] who achieves locally decodable codes of sub-exponential length. For super-constant number of queries, however, better constructions are known. In particular, for $\ell = (\log k)^{O(\frac{1}{p-0.5})}$ Babai et al [1] constructed LDCs of size $K = k^{1+(p-0.5)}$.

We derive following important conclusions from these results: all known constructions in the literature are either exponential in k or the query complexity is a huge polynomial in $\log k$. Furthermore, most of these constructions are able to provide only a *constant* probability of error correction which does not vanish with the size of the message.

Our Results. We consider the construction of locally decodable codes against computationally bounded channel. Under the *minimal* cryptographic assumption that one-way functions exist, we show how to construct *asymptotically good* locally decodable codes over a *binary* alphabet. Notice that small alphabet size is usually a requirement as most channels are best at transmitting only bits. Thus we have achieved locally decodable codes over binary alphabets with constant information rate. This is already much better than all the known constructions in the literature. Our constructions require that the encoding and decoding algorithms share a secret key that is not known to the channel. For this reason we call our codes *private locally decodable codes*.

By reading at most $\ell = \omega(\log^2 \kappa)$ bits in the codeword, our codes can correctly recover any given bit with probability $p \geq 1 - \kappa^{-\omega(1)}$, where κ is the security parameter, as long as the number of errors are less than a suitably chosen (constant) fraction. Thus, the probability of incorrect decoding is $\kappa^{-\omega(1)}$ which is negligible in the security parameter.[3] Furthermore, if we allow the sender and the receiver to share a (synchronized) shared (such as a *public* counter), then our codes can have query complexity only $\omega(\log \kappa)$. We also show that $\ell = \omega(\log \kappa)$ is necessary in order to achieve negligibly small probability of incorrect decoding. Thus, our codes have (almost) optimal query complexity.

Our codes are non-adaptive in nature. That is, the decoding procedure can make all its ℓ queries at once without any dependence on the answers received from the corrupted word. This is a feature that might be desirable in some applications.

In some sense our results are incomparable to previous work because we work only against a computationally bounded adversary. But, a series of lower bound results and the poor information rate of best known constructions from previous work provide good motivation to study the problem in this new (weak yet reasonable) model.

Organization. The rest of this article is organized as follows. The next section presents relevant background from coding theory and cryptography. We then describe our model which is followed by our constructions. We then conclude the paper by noting that the computationally bounded channel model is a promising model where we can go beyond the classical bounds in the theory of error-correcting codes.

[3] We can also choose the length of the input instead of the security parameter and then the error probability will be negligible in the length of the input.

2 Definitions

In this section we will present relevant coding theory and cryptography. When dealing with codes, small alphabet size is usually preferred. Thus, unless specified otherwise, from now onwards we describe our constructions only for binary alphabets. It is straightforward to see their general version that has larger alphabet size. First we present some notations.

NOTATION. Vectors over $\{0,1\}$ will be represented in bold, e.g., $\mathbf{x}, \mathbf{y}$. Because we are working over binary alphabets, occasionally we may refer to vectors over $\{0,1\}$ as (bit) strings. Concatenation of two vectors $\mathbf{x}, \mathbf{y}$ is denoted by $\mathbf{x} \circ \mathbf{y}$. By $[n]$ we denote the set of positive integers smaller than or equal to n: $\{1, 2, \ldots, n\}$. A function $\nu(n)$ is negligible in n if it vanishes faster than the inverse of every polynomial $P(n)$ for a sufficiently large choice of n. Notation $\Delta(\mathbf{x}, \mathbf{y})$ represents the hamming distance between vectors $\mathbf{x}$ and $\mathbf{y}$ which is the number of *alphabet* positions in which they differ. By $\mathbf{x}[j]$ we denote the j^{th} bit of $\mathbf{x}$. If S is a set then the process of selecting an element e from S uniformly at random, is denoted by: $e \xleftarrow{\$} S$. By π we denote a permutation (or a map) which permutes the bits of a given string $\mathbf{x}$ by sending its j^{th} bit to the position $\pi(j)$. We will abuse the notation and denote by $\pi(\mathbf{x})$ the string obtained by applying π to $\mathbf{x}$ as above.

We now present some standard definitions, mostly taken from existing literature, e.g. [19].

Definition 1 (Coding Scheme) *An $(K, k)_q$-coding scheme $\mathcal{C} = (\mathcal{S}, \mathcal{R})$ over the alphabet Σ is a pair of* encoding *and* decoding *functions $\mathcal{S} : \Sigma^k \to \Sigma^K$ and $\mathcal{R} : \Sigma^K \to \Sigma^k$ for some positive integers $K > k, q = |\Sigma| \geq 2$. The (information)* rate *of the scheme, denoted R, is defined as $R = \frac{k}{K}$. The (minimum)* distance *of the coding scheme, denoted δ, is defined as $\delta = \min_{\mathbf{x_1}, \mathbf{x_2} \in \Sigma^k} \Delta(\mathcal{S}(\mathbf{x_1}), \mathcal{S}(\mathbf{x_2}))$*

In this paper we will be interested in asymptotic behavior of our codes. This requires us to consider infinite families of codes. Thus, we augment our current notation by indexing them and redefine the parameters such as the rate of the code.

Definition 2 (Family of Coding Schemes) *Let $\mathbb{C} = \{C_i\}_{i=1}^{\infty}$ be an infinite family of coding schemes where C_i is a $(K_i, k_i)_{q_i}$-coding scheme and $\lim_{i \to \infty} K_i = \infty$.*

The asymptotic information rate and minimum distance of $\mathbb{C}$, denoted $R(\mathbb{C})$ and $\delta(\mathbb{C})$ respectively, are defined as $R(\mathbb{C}) = \liminf_{i \to \infty} k_i/K_i$ and $\delta(\mathbb{C}) = \liminf_{i \to \infty} \delta_i/K_i$.

If $\{\mathcal{S}_i\}$ and $\{\mathcal{R}_i\}$ can be computed by two uniform probabilistic polynomial time algorithms, we say that the coding scheme is efficient.

In our constructions, as the encoding and decoding function do not change with i and only the parameters such as message length, code length etc. vary with i, we will drop the index i from $\mathcal{S}, \mathcal{R}$. Now we turn to the definition of standard locally decodable codes:

Definition 3 (Locally Decodable Code) *An ℓ-locally decodable code over a binary alphabet for error rate ρ and error-correction probability $p > \frac{1}{2}$, abbreviated as (ℓ, ρ, p)-LDC, is a pair of probabilistic algorithms $(\mathcal{S}, \mathcal{R})$, where $\mathcal{S} : \{0,1\}^{k_i} \to \{0,1\}^{K_i}$ and $\mathcal{R}$ are the encoding and decoding algorithms respectively. If $\mathbf{x} \in \{0,1\}^{k_i}$ is the message and $\mathbf{y} \leftarrow \mathcal{S}(\mathbf{x})$ is its encoding then we require that on input $j \in [k_i]$, the algorithm $\mathcal{R}$ reads at most ℓ bits from a given word $\mathbf{y}'$ and outputs a bit b such that $\Pr[b = \mathbf{x}[j]] \geq p$ provided that $\Delta(\mathbf{y}, \mathbf{y}') \leq \rho K_i$ for some constant ρ.*

Notice that locally decodable code is also a coding scheme as defined above but with the exception that the decoding algorithm does not have the whole codeword as input. It rather takes a single bit-position as input and is given oracular access to the codeword. Thus, terms such as a family of locally decodable coding schemes and asymptotic information rate are also defined analogously for locally decodable codes.

3 Our Model

We work in a shared key model where the encoding and decoding algorithms share some small secret information not known to the channel. In particular, this information will be the secret key to the pseudorandom permutation generator.

Deviating from traditional goals, we focus on constructing codes with high probability of recovering any given bit rather than some constant probability larger than 1/2. In particular, we require the probability of *incorrect* decoding to be *negligible* in the message length. Of course small query complexity is desirable too along with negligible probability of incorrect decoding.

Because the encoding and decoding algorithms must share a key in our model, our codes are named private locally decodable codes. We present the definition of a private locally decodable code below.

Definition 4 (Private ℓ-Locally Decodable Code) *Let κ be the security parameter. A private ℓ-locally decodable code for a family of parameters $\{(K_i, k_i)\}_{i=1}^{\infty}$ is a triplet of probabilistic polynomial time algorithms $(\mathcal{K}, \mathcal{S}, \mathcal{R})$ such that:*

- *$\mathcal{K}(1^\kappa)$ is the key generation algorithm that takes as input the security parameter κ and outputs a secret key sk.*
- *$\mathcal{S}(\mathbf{x}, sk)$ is the encoding algorithm that takes as input the message $\mathbf{x}$ of length $k_i = \text{poly}(\kappa)$ and the secret key sk. The algorithm outputs $\mathbf{y} \in \{0,1\}^{K_i}$ that denotes an encoding of $\mathbf{x}$.*
- *$\mathcal{R}(j, sk)$ denotes the decoding algorithm, which takes as input a bit position $j \in [k_i]$ and the secret key sk. It outputs a single bit b denoting the decoding of $\mathbf{x}[j]$ by making at most ℓ (adaptive) queries into a given a codeword $\mathbf{y}'$ possibly different from $\mathbf{y}$.*

The information rate of the scheme is $\liminf_{i\to\infty} k_i/K_i$.

Parameter ℓ is called the query complexity of the code. Notice that in our definition, the decoding algorithm is supposed to have the same secret key sk as was used to encode the message. Obviously this definition does not make sense until we introduce the probability of correctly obtaining $\mathbf{x}[j]$ using the decoding procedure. But before that, we need to explain the game between the channel and the encoding and decoding algorithms.

A computationally bounded adversarial channel $\mathcal{C}$ with error rate ρ is a probabilistic polynomial time algorithm which repeatedly interacts with the encoding algorithm $\mathcal{S}$ and the decoding algorithm $\mathcal{R}$ polynomially many times until it terminates. Each iteration takes place as follows:

1. Given a security parameter κ, the key generation algorithm outputs a secret key $sk \leftarrow \mathcal{K}(1^\kappa)$. The secret is given to both $\mathcal{S}, \mathcal{R}$ but not to the channel. The channel is given κ.
2. In h^{th} iteration, the channel $\mathcal{C}$ chooses a message $\mathbf{x}^{(h)} \in \{0,1\}^{k_i}$ and hands it to the sender.
3. The sender computes $\mathbf{y}^{(h)} \leftarrow \mathcal{S}(\mathbf{x}^{(h)}, sk)$ and hands the codeword $\mathbf{y}^{(h)} \in \{0,1\}^{K_i}$ back to the channel.
4. The channel corrupts at most a fraction ρ of all K_i bits in $\mathbf{y}^{(h)}$ to output the corrupted codeword $\mathbf{y}'^{(h)}$, i.e., $\Delta(\mathbf{y}^{(h)}, \mathbf{y}'^{(h)}) \leq \rho K_i$. It gives $\mathbf{y}'^{(h)}$ and a challenge bit j to the receiver $\mathcal{R}$.
5. The receiver makes at most ℓ (possibly adaptive) queries into the new codeword $\mathbf{y}'^{(h)}$ and outputs $b \leftarrow \mathcal{R}(j, sk)$.

We say that a code $(\mathcal{K}, \mathcal{S}, \mathcal{R})$ *correctly decodes from error rate ρ with high probability* if for all probabilistic polynomial time algorithms $\mathcal{C}$ in the above experiment, for all messages $\mathbf{x} \in \{0,1\}^{k_i}$, and for all $j \in [k_i]$ we have that $\Pr\left[b \neq \mathbf{x}^{(h)}[j]\right] = \nu(\kappa)$, where the probability is taken over the random coins of $\mathcal{K}, \mathcal{S}, \mathcal{R},$ and $\mathcal{C}$.

In above definition, we have that maximum value of h is bounded from above by a value polynomial in the length of the input. If we have a code that only works (i.e., correctly decodes from error rate ρ with high probability) *once* (i.e., only for $h = 1$) and guarantees nothing for repeated executions, we call such a private locally decode to be *one time*.

In the above definition, we assume that the adversary always sends messages of the same length known a priori both to the sender and receiver. We stress that it is merely a *technicality*. If one wants that the adversary be given the flexibility to choose the message lengths, then also our constructions work but with a slight technical modification[4].

4 Our Constructions

In this section we provide our constructions. We do this in two stages. First we provide two constructions which work only once, i.e., they are one-time. First such construction is a simple repetition code with $\log^2 \kappa$ query complexity[5] and the second one is based on any asymptotically good code and has the same query complexity but a better (i.e., asymptotically positive) information rate. In the second stage, we show how to uplift our construction so that we get a code that works for polynomially many invocations, i.e., satisfies our actual definition.

Although we describe our construction for $\log^2 \kappa$ query complexity, they actually work for any query complexity that grows faster than $\log \kappa$, (i.e., $\omega(\log \kappa)$). We also show that $\omega(\log \kappa)$ query complexity is essential if we want decoding error to be negligible in κ. Thus our constructions have optimal query length.

4.1 Constructions for One-time Codes

A Simple Repetition Code. Our first code is a very simple repetition code. Let $\mathbf{x}$ be the string we want to encode. Our repetition code $(\mathcal{K}^{\text{REP}}, \mathcal{S}^{\text{REP}}, \mathcal{R}^{\text{REP}})$ is as follows.

Algorithm $\mathcal{K}^{\text{REP}}(1^\kappa)$ This is a randomized algorithm which simply outputs a truly random permutation π and a truly random mask $\mathbf{r}$ both of size K_i (the code length, to be stated later)[6]. Thus, $sk \leftarrow (\pi, \mathbf{r})$.

Algorithm $\mathcal{S}^{\text{REP}}(\mathbf{x}, sk)$ The algorithm works as follows:

- Compute $\mathbf{x}'$ by repeating each bit of $\mathbf{x}$ for $\log^2 \kappa$ times.
- Compute $\mathbf{y}_1 \leftarrow \pi(\mathbf{x}')$ and output $\mathbf{y} = \mathbf{y}_1 \oplus \mathbf{r}$.

Notice that the size of codeword y is $K_i = k_i \log^2 \kappa$.

[4] Jumping ahead, in our constructions the key generation algorithm needs to output a permutation π and a mask $\mathbf{r}$ truly randomly which are shared by the sender and the receiver as part of their secret keys and their length depends on the message length to be encoded. When the adversary is given the flexibility of choosing message lengths, then our algorithms will instead output a key to a pseudorandom function [9] which will then be used by the encoding and decoding algorithms to compute π and $\mathbf{r}$ pseudorandomly depending upon the message length. Avoiding this technicality renders a clean presentation.

[5] Technically, the query complexity is actually $\lceil \log^2 \kappa \rceil$, but in order to avoid the cluttering in presentation, we shall drop floors and ceiling in formulas. This does not affect our analysis.

[6]

Algorithm $\mathcal{R}^{\text{REP}}(j, sk)$ To decode, the algorithm simply reads all $\ell = \log^2 \kappa$ bit positions of corrupted word $\mathbf{y}'$ that correspond to bit position j of the original message $\mathbf{x}$, and decides by majority after unmasking them with $\mathbf{r}$. Note that computing these bit positions requires reading only ℓ entries from the stored permutation π and hence has *polylogarithmic* running time. The algorithm works as follows:

- Let $j_1, j_2, \ldots, j_\ell$ denote the ℓ bit positions of $\mathbf{x}'$ that have the copies of $\mathbf{x}[j]$. Compute $i_h \leftarrow \pi(j_h)$ for $h = 1, 2, \ldots, \ell$.
- Compute $\mathbf{y}'[i_1] \oplus \mathbf{r}[i_1], \mathbf{y}'[i_2] \oplus \mathbf{r}[i_2], \ldots, \mathbf{y}'[i_\ell] \oplus \mathbf{r}[i_\ell]$ and output the majority bit.

Notice that the query complexity is $\ell = \log^2 \kappa$.

In the above, instead of $\ell = \log^2 \kappa$ we can choose any $\ell = \omega(\log \kappa)$.

Theorem 1 *There exists a constant ρ such that $(\mathcal{K}^{\text{REP}}, \mathcal{S}^{\text{REP}}, \mathcal{R}^{\text{REP}})$ is a* one-time *private $\omega(\log \kappa)$-locally decodable code that correctly decodes from error rate ρ with high probability.*

Proof. It is easy to see that a bit is decoded incorrectly if and only if at least $\lambda = \ell/2$ of its ℓ copies were corrupted. From Lipton's theorem [17], it follows that if the permutation π and the mask $\mathbf{r}$ are truly random, then the adversarial channel $\mathcal{C}$ behaves like a binary symmetric channel which corrupts at most a fraction ρ of all bits. Thus, the probability p of incorrect decoding for a given bit position j can be calculated by a simple combinatorial analysis:

$$p < \frac{\binom{\ell}{\lambda}\binom{n-\lambda}{m-\lambda}}{\binom{n}{m}} < \left(256\frac{\ell}{\lambda} \cdot e^{b+1}\rho\right)^{\lambda} \qquad \text{(see appendix)}$$

which is less than $2^{-\ell} = \nu(k)$ for $\rho = \frac{1}{2^{11}e^{b+1}}$ and $\ell = \omega(\log \kappa)$. Because probability of incorrectly decoding a given bit is negligible and there are only k_i bits, we conclude that probability of incorrectly decoding *any* bit is negligible given that the permutation π and $\mathbf{r}$ are truly random (which is the case). ■

Construction based on Any Asymptotically Good Code. In this section we present the construction of a locally decodable code based on any asymptotically good code. We will present a general construction and its analysis without setting the parameters explicitly. Later on, we will set the parameters suitably so as to obtain a locally decodable code satisfying our goals. We start with the definition of asymptotically good codes.

Definition 5 (Asymptotically Good Codes) *A family of codes $\mathbb{C} = \{C_i\}_{i=1}^{\infty}$ is said to be* asymptotically good *if $R(\mathbb{C}), \delta(\mathbb{C}) > 0$.*

We remark that efficient asymptotically good codes are known [12, 22]. Sometimes we may simply use R and δ and drop the argument $\mathbb{C}$ when it is clear from the context. Also, from now on, in our constructions we will only refer to $C = (\mathcal{S}, \mathcal{R})$ which is a $(A, a)_q$ coding scheme from the family of asymptotically good codes. Let $\frac{1}{\beta}$ be the rate of the code so that $A \leq \beta a$, where β is a constant. Let γ denote the constant fraction such that C can recover from error rate γ (i.e. the number of errors allowed is equal to γA symbols). Because we are working over an alphabet of size q, let $c = \log q$, and we will sometimes say that the message $\mathbf{x}$ is a sequence of $c \cdot a$ *bits* and the codeword $\mathbf{y}$ is a sequence of $c \cdot A$ bits. A symbol is considered corrupted if any of its c bits gets corrupted and hence number of *bit*-errors e from which C can recover is still at most γA.

Our Construction. On a high level, we visualize the message $\mathbf{x}$ as a series of n_i messages each of which will contain a symbols from Σ, or in other words each message will contain a blocks of $c = \log q$ bits each. That is,

$$\mathbf{x} = \overbrace{(\mathbf{x}_1 \circ \mathbf{x}_2 \circ \ldots \circ \mathbf{x}_a)}^{1} \circ \overbrace{(\mathbf{x}_{a+1} \circ \mathbf{x}_{a+2} \circ \ldots \circ \mathbf{x}_{2a})}^{2} \circ \ldots \circ \overbrace{(\mathbf{x}_{(n_i-1)a+1} \circ \mathbf{x}_{(n_i-1)a+2} \circ \ldots \circ \mathbf{x}_{n_i a})}^{n_i}$$

Now each message (contained in parentheses) will be encoded using the encoding function $\mathcal{S}$ of the asymptotically good coding scheme C and all such encodings will be concatenated together. The resulting string will be permuted according to a pseudo-random permutation π and XORed with a pseudorandom string $\mathbf{r}$ to yield the final code. Notice that the message length for our locally decodable code is: $k_i = |\mathbf{x}| = c \cdot a \cdot n_i$. We will choose the parameters A, a for the asymptotically good code C in such a way that we will achieve locally decodable codes with desirable properties. Following is the formal description of our code.

Let C be the asymptotically good (aG) code with $a = \log^2 \kappa$ where κ is the security parameter. Let the rate of the code be $1/\beta$ and error-tolerance γ so that code length $A = \beta a$ and it can correct up to γA errors. Following is the set of algorithms.

Algorithm $\mathcal{K}^{\mathrm{aG}}(1^\kappa)$ Same as for the repetition code: $sk \leftarrow (\pi, \mathbf{r})$

Algorithm $\mathcal{S}^{\mathrm{aG}}(\mathbf{x}, sk)$ The algorithm works as follows:

- Let $\mathbf{x} = \mathbf{w}_1 \circ \mathbf{w}_2 \circ \ldots \circ \mathbf{w}_{n_i}$, where $\mathbf{w}_s = \mathbf{x}_{(s-1)a+1} \circ \mathbf{x}_{(s-1)a+2} \circ \ldots \circ \mathbf{x}_{sa}$ for $s = 1, 2, \ldots, n_i$. Notice that $k_i = ca \cdot n_i$.
- Each $\mathbf{w}_s$ is a sequence of a symbols from Σ. Encode each $\mathbf{w}_s$ using the encoding function $\mathcal{S}$ of C to get encoded words $\mathbf{w}'_s$. That is, for each s, compute:

$$\mathbf{w}'_s \leftarrow \mathcal{S}(\mathbf{w}_s)$$

- Let $\mathbf{x}' = \mathbf{w}'_1 \circ \mathbf{w}'_2 \circ \ldots \circ \mathbf{w}'_{n_i}$. Compute $\mathbf{y}_1 \leftarrow \pi(\mathbf{x}')$ and output $\mathbf{y}_1 \oplus \mathbf{r}$.

Notice that the size of codeword y is $K_i = cAn_i = \frac{A}{a} k_i = \beta k_i$.

Algorithm $\mathcal{R}^{\mathrm{aG}}(j, sk)$ The j^{th} bit of message $\mathbf{x}$ lies in $\mathbf{w}_s$ where $s = \lceil \frac{j}{ac} \rceil$. The decoding algorithm simply reads all the cA bits (corresponding to $\mathbf{w}'_s$) from the (possibly) corrupted encoding $\mathbf{y}'$ using $\ell = cA$ queries, unmasks them using $\mathbf{r}$ and then decodes using the decoding algorithm $\mathcal{R}$ to obtain the complete subsequence $\mathbf{w}_s$. Notice that positions of all cA bits corresponding to $\mathbf{w}'_s$ can be computed using π in sublinear time.

- Let $j_1, j_2, \ldots, j_\ell$ be the bit positions corresponding to the bits of $\mathbf{w}'_s$. Then for all $h = 1, 2, \ldots, \ell$ compute $i_h \leftarrow \pi(j_h)$.
- Compute $\mathbf{y}'[i_1] \oplus \mathbf{r}[i_1], \mathbf{y}'[i_2] \oplus \mathbf{r}[i_2], \ldots, \mathbf{y}'[i_\ell] \oplus \mathbf{r}[i_\ell]$ and obtain points $\mathbf{w}'_s$ (possibly corrupted).
- Apply the decoding algorithm $\mathcal{R}$ instance on possibly corrupted $\mathbf{w}'_s$ to obtain $\mathbf{w}_s$. Output that bit of $\mathbf{w}_s$ which corresponds to the j^{th} bit of $\mathbf{x}$.

Notice that the query complexity is $\ell = cA$.

Above code is a private locally decodable code with *positive* information rate $\liminf_{i \to \infty} \frac{k_i}{K_i} = \frac{1}{\beta}$ and query complexity $\ell = cA = \log q \cdot \beta \log^2 \kappa = O(\log^2 \kappa)$. Notice that instead of using $a = \log^2 \kappa$ it is also possible to use $a = \omega(\log \kappa)$ and then the query complexity would be $\omega(\log \kappa)$ and information rate would still be $\frac{1}{\beta}$. Let us now prove the following.

Theorem 2 *There exists a* constant ρ *such that* $(\mathcal{K}^{\mathrm{aG}}, \mathcal{S}^{\mathrm{aG}}, \mathcal{R}^{\mathrm{aG}})$ *is a* one-time *private* $\omega(\log \kappa)$*-locally decodable code with* constant *information rate that correctly decodes from error rate* ρ *with high probability.*

PROOF. We have already proved the claims about information rate and query complexity. We only need to show that the code indeed correctly recovers from some constant error rate ρ with high probability.

Notice that the algorithm may decode a given bit j incorrectly only if $\mathbf{w}'_s$ is corrupted in at least $\lambda = \gamma A$ bit positions. This is because C can correctly recover from error rate γ. We thus need to bound the probability that more than λ bits of $\mathbf{w}'_s$ are flipped by any adversary. As π and $\mathbf{r}$ are truly random we can use Lipton's theorem, and bound this probability just by analyzing the code represented by $\mathbf{x}'$ in the presence of a binary symmetric channel[7] which only corrupts at most a ρ fraction of all K_i bits. Now the probability p of incorrectly decoding bit j can be bounded by a simple combinatorial argument:

$$p < \frac{\binom{cA}{\lambda}\binom{n-\lambda}{m-\lambda}}{\binom{n}{m}} < \left(256\frac{cA}{\gamma A} \cdot e^{b+1}\rho\right)^{\lambda} \qquad \text{(see appendix)}$$

which is less than $2^{-\ell} = \nu(\kappa)$ for $\rho = \frac{\gamma}{c \cdot 2^{8+\frac{c}{\gamma}} e^{b+1}}$ and $\ell = \omega(\log \kappa)$. Because there are only $k_i = \text{poly}(\kappa)$ bits, it follows that the probability of incorrectly decoding *any* bit is $\nu(\kappa)$. ■

4.2 Final Construction

In this section, we now show how to uplift our one-time constructions so that they work for polynomially many times. Suppose that we had multiple independent permutations π every time we needed to encode a message. In that case, our one-time construction will work for multiple times. But we do not want to keep any synchronized state. So how can we make our constructions work? The idea is to divide the codeword obtained from one-time code into small chunks and then encrypt each chunk using a suitable encryption scheme. This way, we hide the permutation π behind the encryption scheme and hence can use just that permutation every time we encode.

There are two small issues with this. First, we cannot use an off-the shelf encryption scheme because it might blow up the size of the chunk by a factor of the security parameter κ. Thus, we instead encrypt the chunk by a pseudorandom one-time pad obtained from a pseudorandom function [9]. Exact details will be given in the description to follow. Second problem is that we even if a single bit gets corrupted in the ciphertext, the whole ciphertext gets corrupted. Thus, we use an off-the-shelf error-correcting code to further encode the ciphertext obtained from each chunk. We require that this error-correcting code be asymptotically good, otherwise our final code will not have constant information rate. Details follow.

Let f_{KEY} denote a pseudorandom function with the key KEY. Let $(\mathcal{K}^{\text{aG}}, \mathcal{S}^{\text{aG}}, \mathcal{R}^{\text{aG}})$ be the one-time coding scheme that we will use as our base. Let $\log^2 \kappa$ where κ is the security parameter.[8] Our final private locally decodable code $(\mathcal{K}^{\text{FIN}}, \mathcal{S}^{\text{FIN}}, \mathcal{R}^{\text{FIN}})$ is as follows:

Algorithm $\mathcal{K}^{\text{FIN}}(1^\kappa)$ This algorithm first runs $\mathcal{K}^{\text{aG}}((1^\kappa))$ to obtain sk' and then chooses a truly random seed KEY of size $\log^2 \kappa$ for the pseudorandom function f. It sets $sk \leftarrow (sk', \text{KEY})$.

Algorithm $\mathcal{S}^{\text{FIN}}(\mathbf{x}, sk)$ The algorithm works as follows:

- Obtain $y' \leftarrow \mathcal{S}^{\text{aG}}(\mathbf{x}, sk')$. Let $K'_i = |y'|$. Divide y' into chunks $B_1, B_2 \ldots, B_z$ of size a each where $z = K'_i/a$. Now, *encrypt* each B_h as $E_h = (r_h, f_{\text{KEY}}(r_h) \oplus B_h)$ where $h \in [z]$ and r_h is a string of length a chosen uniformly at random.
- Now, *encode* each ciphertext E_h using an *asymptotically good* code of information rate $1/\beta_1$: $F_h \leftarrow \mathcal{S}(E_h)$. Let $y = F_1 \circ F_2 \circ \ldots \circ F_z$. Notice that $|y| = 2\beta_1 K'_i$. Output y.

Notice that the size of codeword y is $K_i = 2\beta_1 K'_i = 2\beta\beta_1 k_i$.

[7] Recall that BSC introduces errors randomly with some fixed error probability.

[8] Any a=$\omega(\log \kappa)$ would also work for our constructions.

Algorithm $\mathcal{R}^{\text{FIN}}(j, sk)$ The decoding algorithm will first run the $\mathcal{R}^{\text{aG}}(j, sk)$ and let $j_1, j_2, \ldots, j'_\ell$ denote the indexes queried by $\mathcal{R}^{\text{aG}}$ (where ℓ' is query length of the one-time code). From the construction, these queries are actually queries into the intermediate code y'. Let B_{j_h}, $0 \leq h \leq \ell$, denote that chunk of y' in which the j_h^{th} bit of y' lies. Then, $\mathcal{R}^{\text{FIN}}$ reads all bits of y' corresponding to each block F_{j_h}, for $j_1, j_2, \ldots, j_\ell$. Thus the query length is $\ell = a\ell'$. Note that these blocks may be corrupted. Now algorithm proceeds as follows:

- Decode each block F_{j_h} using the decoding algorithm $\mathcal{R}$ to obtain (possibly incorrect) blocks $E_{j_h} = (r_{j_h}, E'_{j_h})$. Now compute $B_{j_h} = E'_{j_h} \oplus f_{\text{KEY}}(r_{j_h})$. Notice that B_{j_h} may be totally different from what it was originally when encoded. Read that bit of B_{j_h} that corresponds to j_h^{th} bit of y' and give it to $\mathcal{R}^{\text{aG}}$ when asked.
- Return whatever is returned by $\mathcal{R}^{\text{aG}}$.

Notice that the query complexity is $\ell = a\ell'$.

Above code is a private locally decodable code with *positive* information rate $\liminf_{i\to\infty} \frac{k_i}{K_i} = \frac{1}{2\beta\beta_1}$ and query complexity $\ell = a\ell'$. As, $\ell' = \omega(\log \kappa)$ we could have used any $a = \omega(\log \kappa)$, we have a code with query complexity $\omega(\log^2 \kappa)$ and information rate would still be $\frac{1}{2\beta\beta_1}$. Let us now prove the following.

Theorem 3 *There exists a* constant ρ *such that* $(\mathcal{K}^{\text{FIN}}, \mathcal{S}^{\text{FIN}}, \mathcal{R}^{\text{FIN}})$ *is a private* $\omega(\log^2 \kappa)$*-locally decodable code with* constant *information rate that correctly decodes from error rate* ρ *with high probability.*

Proof (Sketch). We have already proved the claims about information rate and query complexity. We only need to show that the code indeed correctly recovers from some constant error rate ρ with high probability.

We will prove this theorem by a hybrid argument. Consider the game played by the channel with sender and receiver in the definition of correct decoding with high probability. First assume that in this game whenever adversary asks to encode a message, the one-time encoding algorithm always uses a new π and a new mask $\mathbf{r}$ both chosen uniformly at random. We will then show that there exists a constant ρ such that $\mathcal{R}^{\text{FIN}}$) decodes correctly with high probability from error rate ρ.

Let ρ_1 and ρ_2 be the error-tolerance of coding schemes $(\mathcal{K}^{\text{aG}}), \mathcal{S}^{\text{aG}}), \mathcal{R}^{\text{aG}})$ and $(\mathcal{S}, \mathcal{R})$ used by the final code. Now, notice that, channel can corrupt at most ρK_i bits. Because corrupting each block F_h in the final code requires corrupting at least $\rho_2|F_h|$ bits of F_h, the channel can corrupt at most $\frac{\rho K_i}{rho_2|F_h|}$ blocks which cannot be recovered by $\mathcal{R}$. Let S denote the index of these blocks. Now consider an adversary $\mathcal{A}$ for the one-time coding scheme who divides y' into blocks of size a each and corrupts exactly those blocks (by flipping *all* a bits) whose index appears in S. Total bits corrupted by this adversary are $a \cdot |S|$. If $a \cdot |S| \leq \rho_1|y'|$ then $\mathcal{A}$ causes incorrect decoding for the one-time coding scheme whenever the channel causes incorrect decoding for our final code. By setting $\rho = \rho_1\rho_2$ (a constant) we can ensure that $a \cdot |S| \leq \rho_1|y'|$.

Now we turn to prove that above holds even if the encoding algorithm uses a fixed π and $\mathbf{r}$ every time which were chosen once (uniformly at random). We will reduce it to the security of the encryption scheme.

Suppose that the adversary can cause incorrect decoding when a fixed permutation and a fixed mask are used every time. Using a standard hybrid argument it can be shown that there will exist a point (during conversion from one hybrid to another) where the adversary will distinguish whether the permutation used inside is a truly random one or the one that has been used before. But because the message gets encrypted after permuting, by the security of the encryption scheme, all permutations should look the same for a computationally bounded adversary. Thus, if the adversary distinguishes between the two case, it breaks the encryption scheme. We omit the details. ■

4.3 Remarks

First we would like to remark that superlogarithmic query length is essential for correctly decoding with high probability. That is,

Lemma 1 *Private locally decodable codes with query complexity $O(\log \kappa)$ (or smaller) that decode from constant error rate with high probability do not exist.*

PROOF. Let $\ell = O(\log \kappa)$. Following is a very simple adversarial strategy which succeeds in incorrect decoding with non-negligible probability. The channel $\mathcal{C}$ chooses ρK_i bit positions in the encoded word uniformly at random and corrupts them, where ρ is some constant. Now it chooses an index $j \in [k_i]$ uniformly at random and asks us to decode the j^{th} bit of the message. Let us compute the probability of incorrect decoding. Let $i_1, i_2, \ldots, i_\ell$ be the bit positions that the decoding algorithm would have queried. Then probability that the ρK_i corrupted bits include the bit positions $i_1, i_2, \ldots, i_\ell$ is ($\rho K_i > \ell$):

$$\binom{K_i - \ell}{\rho K_i - \ell} / \binom{K_i}{\rho K_i} \approx \rho^\ell = \kappa^{-O(1)}$$

which is non-negligible. ■

Thus, query complexity of our constructions is already optimal up to a logarithmic factor. Furthermore, if we are willing to allow sender and receiver share a small synchronized state then our constructions can achieve the query complexity $\omega(\log \kappa)$ which will be essentially optimal. This is done by choosing the permutation and the mask pseudorandomly based on a shared key for pseudorandom generator and the synchronized state. Details are straightforward and hence omitted.

Usually, the shared state model is not considered an interesting model. However, in our construction, because the state need not be secret and can be implemented by something as simple as a counter, we believe it to be reasonable. In particular it does not require any secret storage or advance sharing of messages that we want to encode!

5 Conclusion

In this paper, we constructed efficient locally decodable codes against a computationally bounded adversarial channel. Our codes can recover any given bit of the message with negligible probability of incorrect decoding and make an optimal number of queries into the corrupted codeword in order to do so. Our results compare favorably to our state of knowledge locally-decodable codes without cryptographic assumptions and are illustrative of the powers of computationally bounded channel model.

References

[1] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *STOC*, pages 21–31, 1991.

[2] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. Bpp has subexponential time simulations unless exptime has publishable proofs. *Computational Complexity*, 3:307–318, 1993.

[3] Amos Beimel and Yuval Ishai. Information-theoretic private information retrieval: A unified construction. In *ICALP*, pages 912–926, 2001.

[4] Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and Jean-François Raymond. Breaking the o(n1/(2k-1)) barrier for information-theoretic private information retrieval. In *FOCS*, pages 261–270, 2002.

[5] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.

[6] Amit Deshpande, Rahul Jain, Telikepalli Kavitha, Jaikumar Radhakrishnan, and Satyanarayana V. Lokam. Better lower bounds for locally decodable codes. In *IEEE Conference on Computational Complexity*, pages 184–193, 2002.

[7] Peter Gemmell, Richard J. Lipton, Ronitt Rubinfeld, Madhu Sudan, and Avi Wigderson. Self-testing/correcting for polynomials and for approximate functions. In *STOC*, pages 32–42, 1991.

[8] Peter Gemmell and Madhu Sudan. Highly resilient correctors for polynomials. *Inf. Process. Lett.*, 43(4):169–174, 1992.

[9] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[10] Oded Goldreich, Howard J. Karloff, Leonard J. Schulman, and Luca Trevisan. Lower bounds for linear locally decodable codes and private information retrieval. In *IEEE Conference on Computational Complexity*, pages 175–183, 2002.

[11] Parikshit Gopalana, Richard J. Lipton, and Y.Z. Ding. Error correction against computationally bounded adversaries. In *Manuscript*, 2004.

[12] Jørn Justesen. A class of constructive asymptotically good algebraic codes. *IEEE Transactions on Information Theory*, 18:652–656, 1972.

[13] Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *STOC*, pages 80–86, 2000.

[14] Iordanis Kerenidis and Ronald de Wolf. Exponential lower bound for 2-query locally decodable codes via a quantum argument. In *STOC*, pages 106–115, 2003.

[15] Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *FOCS*, pages 364–373, 1997.

[16] Michael Langberg. Private codes or succinct random codes that are (almost) perfect. In *FOCS*, pages 325–334, 2004.

[17] Richard J. Lipton. A new approach to information theory. In *STACS*, pages 699–708, 1994.

[18] E Mann. *Private Access to Distributed Information*. 1998. Master's Thesis, Technion.

[19] Silvio Micali, Chris Peikert, Madhu Sudan, and David A. Wilson. Optimal Error Correction Against Computationally Bounded Noise. In *TCC'06*. Springer-Verlag, 2006.

[20] Kenji Obata. Optimal lower bounds for 2-query locally decodable linear codes. In *RANDOM*, pages 39–50, 2002.

[21] Dungjade Shiowattana and Satyanarayana V. Lokam. An optimal lower bound for 2-query locally decodable linear codes. *Inf. Process. Lett.*, 97(6):244–250, 2006.

[22] Michael Sipser and Daniel A. Spielman. Expander codes. In *FOCS*, pages 566–576, 1994.

[23] Adam Smith. Scrambling adversarial errors using few random bits. In *SODA*, 2007.

[24] Madhu Sudan. *Efficient Checking of Polynomials and Proofs and the Hardness of Approximation Problems*. 1992. PhD Thesis, University of California at Berkley.

[25] Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. Pseudorandom generators without the xor lemma (extended abstract). In *STOC*, pages 537–546, 1999.

[26] Stephanie Wehner and Ronald de Wolf. Improved lower bounds for locally decodable codes and private information retrieval. In *ICALP*, pages 1424–1436, 2005.

[27] David Woodruff. New lower bounds for general locally decodable codes. In *ECCC TR07-006*, 2007.

[28] Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. In *STOC*, 2007. Also appears on ECCC as TR06-127 under a different title.

A Probability of incorrect decoding

Here we provide calculations that we used in our proofs for bounding the probability of incorrect decoding. Let n be the length of the code and $m = n\rho$ be the number of errors. In the following we assume that ℓ and λ are quite small ((e.g. polylogarithmic in n)) compared to n and $\ell < \lambda$. First notice that,

$$\binom{\ell}{\lambda} \leq \left(\frac{\ell e}{\lambda}\right)^{\lambda}$$

where e is Euler's constant. Next we have the following using Stirling's approximation:

$$\begin{aligned}
\binom{n-\lambda}{m-\lambda} &\leq 16 \cdot \left(\frac{n-\lambda}{n-m}\right)^{n} \left(\frac{n-m}{m-\lambda}\right)^{m} \left(\frac{m-\lambda}{n-\lambda}\right)^{\lambda} \\
&= 16 \cdot \left(\frac{1}{1-a\rho}\right)^{n} \left(\frac{1-a\rho}{a\rho}\right)^{m} (a\rho)^{\lambda} \quad \text{where } a = \frac{1-\frac{\lambda}{m}}{1-\frac{\lambda}{n}} < 1 \\
&= 16 \cdot S_1 \cdot (a\rho)^{\lambda} \quad \text{where } S_1 = \left(\frac{1}{1-a\rho}\right)^{n} \left(\frac{1-a\rho}{a\rho}\right)^{m}
\end{aligned}$$

Similarly,

$$\binom{n}{m} \geq \frac{1}{16} \cdot S_2 \cdot \quad \text{where } S_2 = \left(\frac{1}{1-\rho}\right)^{n} \left(\frac{1-\rho}{\rho}\right)^{m}$$

Now notice that,

$$\begin{aligned}
\frac{S_1}{S_2} &= a^{-m} \left(\frac{1-\rho}{1-a\rho}\right)^{n-m} < a^{-m} \quad (\text{as } a < 1) = \left(\frac{1-\frac{\lambda}{n}}{1-\frac{\lambda}{m}}\right)^{m} \\
&= \left(\frac{1-\rho\frac{\lambda}{m}}{1-\frac{\lambda}{m}}\right)^{m} = \left(1 + \frac{(1-\rho)\frac{\lambda}{m}}{1-\frac{\lambda}{m}}\right)^{m} = \left(1+\frac{1}{x}\right)^{m} \quad \text{where } \frac{1}{x} = \frac{1-\rho}{\frac{m}{\lambda}-1} \\
&= \left(1+\frac{1}{x}\right)^{xy} \quad \text{where } y = \frac{m}{x} < b\lambda \quad (\text{for some constant } b) \\
&< e^{b\lambda}
\end{aligned}$$

Finally, using all the above relations we get

$$p < \frac{\binom{\ell}{\lambda}\binom{n-\lambda}{m-\lambda}}{\binom{n}{m}} < 256 \left(\frac{\ell e}{\lambda}\right)^{\lambda} (a\rho)^{\lambda} e^{b\lambda} < \left(256\frac{\ell}{\lambda} \cdot e^{b+1}\rho\right)^{\lambda} \quad \text{as } a < 1$$