Private Locally Decodable Codes

Rafail Ostrovsky Omkant Pandey Amit Sahai

Department of Computer Science
University of California, Los Angeles 90024

{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 *constant* information rate and *low* (in fact, optimal) query complexity which can *uniquely* decode any given bit of the message from constant-fraction 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.

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_i symbols from alphabet Σ_1 and the codeword is represented as a sequence of K_i symbols from (a possibly different) alphabet Σ_2. The encoding function is denoted by $\mathcal{S} : \Sigma_1^{k_i} \to \Sigma_2^{K_i}$ and the decoding function is denoted by $\mathcal{R} : \Sigma_2^{K_i} \to \Sigma_1^{k_i}$. The *information rate* (or simply *rate*) of the code is k_i/K_i and measures the amount of extra information needed by the code for correctly decoding from errors.

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). They were explicitly discussed most notably in [1, 27] in the contexts of probabilistically checkable proofs. They were explicitly defined by Katz and Trevisan in [15].

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 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 $(|\Sigma_1|, |\Sigma_2|)$ are desirable too, and indeed 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 [9, 10], probabilistically checkable proof systems [1], worst-case to average-case hardness reductions in the constructions of pseudo-random generators [2, 28], and so on. In cryptography they have been useful due to their interesting connection with private information retrieval protocols [6, 18, 22, 3]. Their interesting properties make them applicable in several database applications such as fault-tolerant data storage [15]. 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*[1] locally decodable codes are not known to exist in this model.

An interesting idea due to Lipton [20], 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 [12] 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 [19] and Smith [26]. 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_i and the message length k_i for (ℓ, ρ, p)-LDCs. Katz and Trevisan [15] first started investigating this direction and showed that for non-adaptive LDCs, K_i is at least $k_i^{1+\frac{1}{\ell-1}}$ (suppressing the dependence on ρ and p). Deshpande et al [7] showed that this bound holds even for adaptive LDCs. These are still the best known lower bounds for general LDCs. Thereafter, a series of papers [11, 24, 25, 16, 29] concentrated on LDCs with $\ell = 2$ (or 3) and established exponential lower bounds. In particular for 2-query LDCs $K_i = \exp(\Omega(\frac{\rho}{2-2p} k_i))$.

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

The other direction focussed on constructing the locally decodable codes. Important constructions in this direction for *constant* query length appeared in [3, 4]. The best known construction is due to Yekhanin [30]. Unfortunately, all these constructions yield codes that are exponentially long in k_i. For super-constant number of queries, however, better constructions are known. In particular, for $\ell = (\log k_i)^{O(\frac{1}{p-0.5})}$ Babai et al [1] constructed LDCs of size $K_i = k_i^{1+(p-0.5)}$.

We derive following important conclusions from these results: all known constructions in the literature are either exponential in k_i or the query complexity is a huge polynomial in $\log k_i$. 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 locally decodable codes with $K_i = O(k_i)$ over a *binary* alphabet with a fairly *small* constant hidden inside the O-notation. 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*.

Our codes can correctly recover any given bit with probability $p \geq 1 - 2^{-O(\ell)}$, where ℓ is the query complexity chosen beforehand, as long as the number of errors are less than a suitably chosen (constant) fraction. Thus, by setting $\ell = \omega(\log k_i)$, we achieve the probability of incorrect decoding to be $k_i^{-\omega(1)}$ which is negligible in k_i. Furthermore, we show that $\ell = \omega(\log k_i)$ is necessary in order to achieve negligibly small probability of incorrect decoding. Thus, our codes have 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.

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 provide two constructions of locally decodable codes in this section and prove the optimality of our results. 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.

2 Preliminaries

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 $\mathbf{y}$. 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: $e \xleftarrow{\$} S$.

Standard locally decodable codes are defined as follows:

Definition 1 (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 $\mathcal{S} : \{0,1\}^{k_i} \rightarrow \{0,1\}^{K_i}$ 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.*

We now turn our attention to pseudo-random permutations π that permute the bits of a given input $\mathbf{x}$. By $\pi(\mathbf{x})$ we shall denote the process of permuting bits of $\mathbf{x}$ according to π and outputting the resulting string. Assuming that the sender and the receiver share some common randomness sk (sometimes called the secret key), such permutations π and their inverses $\overline{\pi}$ can be constructed from pseudorandom generators [5], which in turn can be constructed from any one way function [14] (The assumption that one-way functions exist, is the minimal cryptographic assumption).

The pseudorandom generator takes a small secret randomness $sk \xleftarrow{\$} \{0,1\}^{\kappa_i}$, where κ_i is a security parameter, and converts it into a polynomially long pseudorandom bit sequence. Thus, it provides a common source of randomness to the parties who share sk. Using this randomness we can easily choose a pseudorandom permutation π that permutes the bits of a given input.

Informally, pseudorandom permutations have the property that for all probabilistic polynomial time algorithms $\mathcal{A}$, it is hard to distinguish the operation of pseudorandom permutations from the operation of uniformly random ones. That is, for all bit sequences $\mathbf{x}$, if π_1 is a pseudorandom permutation and π_2 is a truly random permutation then for all efficient algorithms $\mathcal{A}$:

$$|\Pr[\mathbf{y} \leftarrow \pi_1(\mathbf{x}); \mathcal{A}(\mathbf{y}) = 1] - \Pr[\mathbf{y} \leftarrow \pi_2(\mathbf{x}); \mathcal{A}(\mathbf{y}) = 1]| = \nu(\kappa_i)$$

For notation, by $\pi_{sk}^{(K_i)}$ we shall denote the pseudorandom permutation output by some pseudorandom permutation generator algorithm PRP that takes as input a secret randomness sk and message length K_i: $\pi_{sk}^{(K_i)} \leftarrow \mathrm{PRP}(sk, K_i)$. The secret randomness sk is output by a key generation algorithm depending upon a security parameter κ_i: $sk \leftarrow \mathcal{K}(1^{\kappa_i})$. We require that $K_i = \mathrm{poly}(\kappa_i)$. Denote by $\tilde{\pi}_{sk}^{(K_i)}$, modification of the permuting algorithm $\pi_{sk}^{(K_i)}$ that takes as input a bit position $j \in [K_i]$ and outputs its permuted position $j' \in [K_i]$ according to $\pi_{sk}^{(K_i)}$).

Remark 1 *The pseudorandom permutations described here should not be confused with pseudorandom permutations introduced by Luby and Rackoff [21]. Our permutations permute the bits of a given input, whereas permutations in [21] are 1-1 and onto functions that do not necessarily permute the bits of the input.*

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 2 (Private ℓ-Locally Decodable Code) *Let κ_i 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:*

- *$\mathcal{K}(1^{\kappa_i})$ is the key generation algorithm that takes as input the security parameter κ_i 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 $k_i = \text{poly}(\kappa_i)$ 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 also 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 interacts with the encoding algorithm $\mathcal{S}$ and the decoding algorithm $\mathcal{R}$ as follows:

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

We say that a code $(\mathcal{K}, \mathcal{S}, \mathcal{R})$ *uniquely decodes from error rate ρ* 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 $\Pr[b \neq \mathbf{x}[j]] = \nu(k_i)$, where the probability is taken over the random coins of $\mathcal{K}, \mathcal{S}, \mathcal{R},$ and $\mathcal{C}$.

Later on, we consider a stronger definition where the channel is allowed to repeatedly make many encoding requests adaptively before it actually causes an incorrect decoding. We show that with a slight modification, our constructions work against such a stronger adversarial channel too. In fact all we need to do is keep a small state information (such as a counter) that helps in choosing a fresh pseudorandom permutation for every encoding request. Other than this, all construction details remain just the same. Due to the space limitations, these details are given in the appendix.

4 Our Constructions

In this section we will give various constructions of private locally decodable codes. We will start with a simple repetition code with $2\log^2 k_i$ query complexity.[2] Later we will provide codes with

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

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

4.1 A Simple Repetition Code

Let $\mathbf{x}$ be the string we want to encode. Let $\overline{\mathbf{x}}$ denote the string obtained by flipping each bit of $\mathbf{x}$. Let PRP be a pseudorandom permutation with the key selecting algorithm $\mathcal{K}^{\text{PRP}}$. Our repetition code $(\mathcal{K}^{\text{REP}}, \mathcal{S}^{\text{REP}}, \mathcal{R}^{\text{REP}})$ is as follows.

Algorithm $\mathcal{K}^{\text{REP}}(1^{\kappa_i})$ This is just the $\mathcal{K}^{\text{PRP}}()$ algorithm that outputs the secret randomness $sk \leftarrow \mathcal{K}^{\text{PRP}}(1^{\kappa_i})$.

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

- Compute $\mathbf{x}'$ by repeating each bit of $\mathbf{x}$ for $2\log^2 k_i$ times, where $k_i = |\mathbf{x}|$. Now let $\mathbf{x}'' = \mathbf{x}' \circ \overline{\mathbf{x}'}$. Notice that $K_i = |\mathbf{x}''| = k_i \cdot 4\log^2 k_i$.
- Let $\pi_{sk}^{(K_i)} \leftarrow \text{PRP}(sk, K_i)$. Compute and output $\mathbf{y} \leftarrow \pi_{sk}^{(K_i)}(\mathbf{x}'')$.

Notice that the size of codeword y is $K_i = 4k_i \log^2 k_i$.

Algorithm $\mathcal{R}^{\text{REP}}(j, sk)$ To decode, the algorithm simply reads all $\ell = 2\log^2 k_i$ bit positions of $\mathbf{y}$ that correspond to bit position j of the original message $\mathbf{x}$, and decides by majority. Computing these bit positions can be done easily by the restricted algorithm $\tilde{\pi}_{sk}^{(K_i)}$. 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 $i_h \leftarrow \tilde{\pi}_{sk}^{(K_i)}(j_h)$ for $h = 1, 2, \ldots, \ell$.
- Read $\mathbf{y}[i_1], \mathbf{y}[i_2], \ldots, \mathbf{y}[i_\ell]$ and output the majority bit.

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

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

Proof. It is easy to see that a bit is decoded incorrectly if and only if at least half of its $\ell = 2\log^2 k_i$ copies were corrupted. Assuming that the permutation $\pi_{sk}^{(K_i)}$ is truly random, the probability of incorrect decoding for a given bit position j is thus:

$$\Pr[j^{\text{th}} \text{ bit decodes incorrectly}] = \sum_{h=\log^2 k_i}^{2\log^2 k_i} \binom{2\log^2 k_i}{h} p^h (1-p)^{2\log^2 k_i - h}$$

where p is the probability that a given copy of the original bit is corrupted. Now notice that total number of corrupted bits is ρK_i. Notice that the final codeword $\mathbf{y}$ contained $K_i/2$ 0s and the same number of 1s. Thus probability that any given bit of $\mathbf{x}'$ gets corrupted is less than $\frac{\rho K_i}{K_i/2} = 2\rho$. Thus, we conclude that $0 \le p \le 2\rho$. Now setting $\rho = 1/34$, noticing that the binomial coefficient $\binom{n}{r}$ is maximum when $r = n/2$, and using Stirling's approximation we conclude that:

$$\Pr[j^{\text{th}} \text{ bit decodes incorrectly}] < \left(\frac{4 \cdot 2^{2\log^2 k_i}}{\sqrt{\pi}\log k_i} \cdot \left(\frac{p}{1-p} \right)^{\log^2 k_i} \cdot 1 \right) \times \log^2 k_i = \nu(k_i)$$

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 $\pi_{sk}^{(K_i)}$ of $\mathbf{x}''$ is truly random. We now show that the probability of incorrect decoding remains negligible in k_i even if $\pi_{sk}^{(K_i)}$ is pseudorandom.

To prove this, suppose that the probability of incorrect decoding is $\hat{p}$ when $\pi_{sk}^{(K_i)}$ is pseudorandom. We show that if $\hat{p}$ is not negligible then we can compromise the security of the pseudorandom permutation $\pi_{sk}^{(K_i)}$. To do so, we construct an adversary $\mathcal{A}$ which can decide, with non-negligible probability, whether a given string is being permuted according to a pseudorandom permutation just by querying the challenge permutation once. The adversary $\mathcal{A}$ is given access to the challenge oracle which will permute a given input string according to a permutation (either pseudorandom or random).

The adversary works as follows. On input the security parameter κ_i, it instantiates our locally decodable code with κ_i and gives κ_i to the channel $\mathcal{C}$. When the channel gives a message $\mathbf{x}$ for encoding, $\mathcal{A}$ computes the intermediate message $\mathbf{x}''$ according to the encoding algorithm $\mathcal{S}^{\mathrm{RM}}$ but does not permute it. Instead, it asks the permuting oracle to permute $\mathbf{x}''$ and receives the answer, say $\mathbf{y}$. It gives $\mathbf{y}$ to $\mathcal{C}$ and receives the corrupted codeword $\mathbf{y}'$ and a challenge bit position j. Now A performs the role of the R^{RM} and decodes the j^{th} bit according to the normal decoding procedure. $\mathcal{A}$ outputs 0 if the decoding is correct and 1 otherwise.

Standard probability calculations show that the probability of $\mathcal{A}$'s success is at least $|\hat{p} - \nu(k_i)|$, which is non-negligible unless $\hat{p}$ is also negligible in k_i. Hence the theorem. ■

4.2 Construction based on Reed-Muller codes

In this section we present the construction of a locally decodable code based on Reed-Muller codes. 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 a review of Reed-Muller codes.

Reed-Muller Codes. In Reed-Muller codes, the message $\mathbf{x}$ defines an m-variate polynomial $P(t_1, t_2, \ldots, t_m)$ of degree d over a finite field $\mathbb{F}_q$. Here, the message is considered as a sequence of a *symbols* from $\mathbb{F}_q$. As we are working with binary alphabets, we will sometimes say that the message m is a sequence of $c \cdot a$ bits where $c = \log q$. The message is encoded by evaluating the polynomial P at suitably chosen A points in $\mathbb{F}_q^m$. Once again notice that the encoded message will have $c \cdot A$ bits. Such a code is called an $(A, a)_q$-Reed-Muller code. Parameters m, q, d, a, and A are all related with each other in certain fashion. It is immediately obvious that $q^m \geq A > a$.

For the sake of concreteness, we consider a specific instance of Reed-Muller codes which will be helpful in understanding our further explanations. Let q be a *constant* and a prime power (e.g., $q = 2^5$). Let the degree $d < q$ also be a constant. In such a case, the message length is: $a = \binom{m+d}{m}$ symbols from $\mathbb{F}_q$. Let $A = \beta a$ for some constant β and let $u_1, u_2, \ldots, u_A$ be (suitably chosen) points in $\mathbb{F}_q^m$. The parameter m will be chosen later on and we will ensure that $q^m > A$. The encoding of $\mathbf{x}$ is the evaluation of polynomial $P(t_1, t_2, \ldots, t_m)$ at points $u_1, u_2, \ldots, u_A$ where P is defined by $\mathbf{x}$. The minimum distance of this code is $(1 - \frac{d}{q})A$ symbols[3] and it can correct errors $e \leq \alpha(1 - \frac{d}{q})A$ where $\alpha < 1/2$ is a positive constant. We will refer to this specific instance of Reed-Muller codes as the RM-instance.

[3]In this description, as the construction is over a field of q elements, the minimum distance means that each code differs from every other code in at least $(1 - \frac{d}{q})$ symbols. But originally we are working on bits (field of 2 elements), thus for us the distance would be $(1 - \frac{d}{q})$ bits and a symbol (or point) will be considered corrupted if any of its c bits gets corrupted.

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 $\mathbb{F}_q$, 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 RM-instance of Reed-Muller code and all such encodings will be concatenated together. The resulting string will be concatenated with its complement and then permuted according to a pseudo-random permutation π to yield the final code. Notice that $k_i = |\mathbf{x}| = c \cdot a \cdot n_i$. Later on, we will choose m in the RM-instance in such a way that we will achieve locally decodable codes with desirable properties. Following is the formal description of our code.

In the RM-instance, select m to be such that $a = \binom{m+d}{m} = \log^2 k_i$ where k_i is the length of the message that we want to encode using our *private locally decodable code*. It can be done, for example, by choosing $d = 2, m = O(\log k_i)$. Let κ_i be the security parameter. Other quantities implicitly defined by RM are $q, \alpha, \beta, A, \mathbb{F}_q$ etc. Following is the set of algorithms.

Algorithm $\mathcal{K}^{\mathrm{RM}}(1^{\kappa_i})$ This is just the $\mathcal{K}^{\mathrm{PRP}}()$ algorithm that outputs the secret randomness $sk \leftarrow \mathcal{K}^{\mathrm{PRP}}(1^{\kappa_i})$.

Algorithm $\mathcal{S}^{\mathrm{RM}}(\mathbf{x}, sk)$ 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 $\mathbb{F}_q$. Encode each $\mathbf{w}_s$ using the RM-instance to get encoded words $\mathbf{w}'_s$. That is, for each s, compute:

$$\mathbf{w}'_s = P_s(u_1) \circ P_s(u_2) \circ \ldots \circ P_s(u_A)$$

- Let $\mathbf{x}' = \mathbf{w}'_1 \circ \mathbf{w}'_2 \circ \ldots \circ \mathbf{w}'_{n_i}$. Compute $\mathbf{x}'' = \mathbf{x}' \circ \overline{\mathbf{x}'}$. Let $K_i = |\mathbf{x}''|$.
- Let $\pi_{sk}^{(K_i)} \leftarrow \mathrm{PRP}(sk)$. Compute and output $\mathbf{y} \leftarrow \pi_{sk}^{(K_i)}(\mathbf{x}'')$.

Notice that the size of codeword y is $K_i = cAn_i \cdot 2 = \frac{2A}{a} k_i = 2\beta k_i$.

Algorithm $\mathcal{R}^{\mathrm{RM}}(j, sk)$ The j^{th} bit of message $\mathbf{x}$ lies in $\mathbf{w}_s$ where $s = \lceil \frac{j}{ac} \rceil$. The corresponding polynomial is thus P_s. The decoding algorithm simply reads all the A points of P_s from the (possibly) corrupted encoding $\mathbf{y}'$ using $\ell = cA$ queries and decodes using the normal Reed-Muller decoding algorithm to obtain the complete subsequence $\mathbf{w}_s$. Notice that positions of all cA bits corresponding to the A points of P_s can be computed using $\tilde{\pi}_{sk}^{(K_i)}$.

- 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 $i_h \leftarrow \tilde{\pi}_{sk}^{(K_i)}(j_h)$.
- Read $\mathbf{y}'[i_1], \mathbf{y}'[i_2], \ldots, \mathbf{y}'[i_\ell]$ and obtain points $P_s(u_1), P_s(u_2), \ldots, P_s(u_A)$ some of which may be corrupted and hence may not lie on the polynomial P_s.
- Apply the decoding algorithm of RM-instance on these points 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 *constant* information rate $\frac{k_i}{K_i} = \frac{1}{2\beta}$ and query complexity $\ell = cA = \log q \cdot \beta \log^2 k_i = O(\log^2 k_i)$. Notice that instead of using $a = \log^2 k_i$ it is also possible to use $a = \omega(\log k_i)$ and then the query complexity would be $\omega(\log k_i)$ and information rate would still be $\frac{1}{2\beta}$. Let us now prove the following.

Theorem 2 *There exists a* constant ρ *such that* $(\mathcal{K}^{\mathrm{RM}}, \mathcal{S}^{\mathrm{RM}}, \mathcal{R}^{\mathrm{RM}})$ *is a* $\omega(\log k_i)$*-private locally decodable code with* constant *information rate that* uniquely *decodes from error rate* ρ*.*

Proof. We have already proved the claims about information rate and query complexity. We only need to show that the code indeed uniquely recovers from some constant error rate ρ.

Let us recall the construction of our locally decodable code $\mathbf{y}$ for message $\mathbf{x}$ with the RM-instance of Reed-Muller codes. The code $\mathbf{y}$ was a permutation of $\mathbf{x}'' = \mathbf{x}' \circ \overline{\mathbf{x}'}$. Furthermore $\mathbf{x}'$ contained A points on each polynomial P_s for $s = 1, 2, \ldots, n_i$. Because RM-instance can correct up to $e \leq \alpha(1 - \frac{d}{q})A$ symbol errors, any given bit j of message $\mathbf{x}$ can decode incorrectly only when more than $\alpha(1 - \frac{d}{q})A$ points of polynomial $P_{\lceil \frac{j}{ca} \rceil}$ out of its A points become corrupted. For convenience, let $\gamma = \alpha(1 - \frac{d}{q})$ and $\lambda = \alpha(1 - \frac{d}{q})A = \gamma A$.

Thus, in order to calculate the probability of incorrect decoding we need to calculate the probability that more than λ points of any given polynomial P_s will be corrupted. Consider the A points of any given polynomial P_s in the encoding: $P_s(u_1), P_s(u_2), \ldots, P_s(u_A)$. Any of these points, say $P_s(u_1)$, gets corrupted if any of its $c = \log q$ bits gets flipped. We know that the corrupted word $\mathbf{y}'$ is such that $\Delta(\mathbf{y}, \mathbf{y}') \leq \rho K_i$. Because there are equal number of 0s and 1s in $\mathbf{y}$, probability that any *given* bit of $P_s(u_1)$ is corrupted in $\mathbf{y}'$ is less than $\frac{\rho K_i}{K_i/2} = 2\rho$ assuming that the permutation $\pi_{sk}^{(K_i)}$ to be truly random. Thus, the probability of $P_s(u_1)$ being good is more than $(1 - 2\rho)^c$. Hence,

$$p = \Pr\left[\text{A given point of } P_s \text{ is corrupted}\right] \leq 1 - (1 - 2\rho)^c$$

Now, the probability that a given bit j of the message is corrupted, is equal the probability that at least λ points of $P_{\lceil \frac{j}{ca} \rceil}$ are corrupted. This can be upper bounded by:

$$\Pr\left[j^{\text{th}} \text{ bit decodes incorrectly}\right] < \sum_{e=\lambda}^{A} \binom{A}{e} p^e (1-p)^{A-e}$$

As we did in the proof of repetition code, using Striling's approximation this value can be upper bounded by:

$$\sqrt{\frac{32A}{\pi}} \left(2\left(\frac{p}{1-p}\right)^{\gamma}\right)^{A} \leq \sqrt{\frac{2\beta \log^2 k_i}{\pi}} \left(2\left(\frac{1}{(1-2\rho)^c} - 1\right)^{\gamma}\right)^{\beta \log^2 k_i}$$

which is negligible in k_i if we set $\left(2\left(\frac{p}{1-p}\right)^{\gamma}\right)^{\beta} = \frac{1}{2}$. Which in turn implies that for an appropriate *constant* ρ the probability of incorrect decoding is negligible in k_i. Furthermore, notice that this claim remains true for any $A = \omega(\log k_i)$. Because there are only polynomially many bits, it follows that probability of incorrectly decoding *any* bit is $\nu(k_i)$ if the permutation $\pi_{sk}^{(K_i)}$ is truly random.

From here on, using exactly the ideas used in the proof of repetition code, we conclude that the probability of incorrect decoding is negligible in k_i for this code. Thus, the code uniquely decodes from error rate ρ. ■

4.3 Two Remarks

Here we would like to remark two important observations. Firstly we show that the query length in our constructions is optimal. That is,

Lemma 1 *Private locally decodable codes for parameters* (K_i, k_i) *and with query complexity* $O(\log k_i)$ *(or smaller) that uniquely decode from constant error rate do not exist.*

PROOF. Let $\ell = O(\log k_i)$. 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 ($\rho K_i > \ell$):

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

which is non-negligible. ■

Next, we want to remark about our Reed-Muller construction. Why did we not work with Reed-Solomon codes instead of Reed-Muller? This is because an analogous construction with Reed-Solomon code would not have worked. The problem comes with the field size. For the decoding error to be negligible in k_i, we need $A = a\omega(\log k_i)$. This requires that the field size be $q > \omega(\log k_i)$ so that we will have enough number of points to evaluate the polynomial at. But this q is not a constant anymore and this messes up the probability calculations. Reed-Muller codes instead allow us to vary the number of variables in the polynomials and thus it was possible to base our construction on Reed-Muller codes.

Because the field size is the only reason why we could not base our construction on Reed-Solomon codes, it seems that algebraic geometric (AG) codes [13] can help. Indeed, it is possible to base our construction on AG-codes. In fact the construction based on AG-codes is very similar to our Reed-Muller based construction and a rough sketch is given in the appendix.

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] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo random bits. In *FOCS*, pages 112–117, 1982.

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

[7] 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.

[8] A. Garcia and H. Stichtenoth. A tower of artin-schreier extensions of function fields attaining the drinfelf-vladut bound. *Invent. Math.*, 121:211–222, 1995.

[9] 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.

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

[11] 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.

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

[13] V.D. Goppa. Algebraic geometric codes. *Math. USSR-Izv*, 21(1):75–93, 1983.

[14] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

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

[16] 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.

[17] Ravi Kumar, Sridhar Rajagopalan, and Amit Sahai. Coding constructions for blacklisting problems without computational assumptions. In *CRYPTO*, pages 609–623, 1999.

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

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

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

[21] Michael Luby and Charles Rackoff. Pseudo-random permutation generators and cryptographic composition. In *STOC*, pages 356–363, 1986.

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

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

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

[25] 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.

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

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

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

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

[30] Sergey Yekhanin. New locally decodable codes and private information retrieval schemes. In *ECCC TR06-127*, 2007.

A Construction based on Algebraic Geometric Codes

Let us provide a brief sketch of a construction based on AG-codes. Recall that our construction based on Reed-Muller codes divides the message into smaller subsequences of size a each and then encodes each one of them by a properly chosen instance (the RM-instance) of Reed-Muller codes. All such encodings are then concatenated together and the resulting string is again concatenated with its complement. The string so received is then permuted according to a pseudorandom permutation to yield encoding. The construction based on AG-codes is exactly the same except that the RM-instance will now be replaced by a suitable instance of AG-codes in order to encode each subsequence. Thus, we only need to sketch a proper instance of AG-codes and argue that it gives us a private locally decodable code with the desired properties.

First let us recall how AG-codes (also known as Goppa codes [13]) are defined. Following [17], let $K/\mathbb{F}_q$ be an algebraic function field of one variable with field of constants $\mathbb{F}_q$ and genus g. Let us denote the places of degree one of K by $\mathbb{P}(K)$. As in the case of RM-instance, we will need an AG-code with constant information rate and hence we choose $A = \beta a$ for some constant β. As a consequence we need $|\mathbb{P}(K)| > A + 1$. For $\mu < A$ and Q a place of degree one, let $L(\mu Q)$ denote the set of all $f \in K$ which have no poles except one at Q of order at most μ. Then,

Definition 3 (AG codes) *Let $Q, u_1, u_2, \ldots, u_A \in \mathbb{P}(K)$ be distinct. The AG code $C(\mu Q; u_1, u_2, \ldots, u_A)$ is given by the subspace $\{f(u_1), f(u_2), \ldots, f(u_A) | f \in L(\mu Q)\}$.*

The minimum distance of the code is at least $A - \mu$ and the message length is at least $\mu - g + 1$. Thus, by carefully choosing parameters μ, g we can have $a = \log^2 k_i$ and $A = \beta a$ for constant β. In particular, using the methods of Garcia-Stichtenoth [8] we can easily construct a function field by F_A by extending $\mathbb{F}_{q^2}$ so that it will have enough places of degree one and an appropriate genus. We refer the reader to [8, 17] for complete details.

Once we have such an AG-code, we can see that our private locally decodable codes will have to read $\{f(u_1), f(u_2), \ldots, f(u_A)\}$ in order to decode. Thus query complexity will be $cA = \omega(\log k_i))$ and the information rate will be constant. The same analysis will also work out for unique decoding and for some constant ρ as the size of field $\mathbb{F}_q$ is constant.

B The Definition of Unique Decoding

We defined the unique decoding via a game between the channel and the encoding and decoding algorithms. However, our game did not allow the channel to access the encoding algorithm multiple

times before actually attempting to cause a decoding error. Here we present a game that allows the adversary to adaptively make several attempts in order to cause decoding error.

In this setting however, we allow the sender and the receiver to share a synchronized state st. Like in previous works [23], we require that the channel does not drop or reorder the messages. The state need not be secret from the channel and usually a simple counter would be enough for it. The encoding and decoding algorithms should be able to update the state information by themselves during multiple interactions. We now present the security game below.

The channel $\mathcal{C}$ interacts with the sender $\mathcal{S}$ and receiver $\mathcal{R}$ repeatedly until it terminates. Each iteration of interaction takes place as follows:

1. 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.
2. The sender computes $(\mathbf{y}^{(h)}, st^{(h+1)}) \leftarrow \mathcal{S}(\mathbf{x}^{(h)}, sk, st^{(h)})$ and hands the codeword $\mathbf{y}^{(h)} \in \{0,1\}^{K_i}$ back to the channel.
3. 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 $\mathcal{R}$.
4. The receiver makes at most ℓ (adaptive) queries into the new codeword $\mathbf{y}'^{(h)}$ and outputs $(b, st^{(h+1)}) \leftarrow \mathcal{R}(j, sk, st^{(h)})$.

We say that a code $(\mathcal{K}, \mathcal{S}, \mathcal{R})$ *uniquely decodes from error rate* ρ 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 $\Pr\left[b \neq \mathbf{x}^{(h)}[j]\right] = \nu(k_i)$, where the probability is taken over the random coins of $\mathcal{K}, \mathcal{S}, \mathcal{R},$ and $\mathcal{C}$.

We now argue that the constructions presented in this paper achieve the claimed parameters even under this definition as long as the state information shared by $\mathcal{S}$ and $\mathcal{R}$ remains synchronized. This is because, first notice that $\mathcal{S}$ and $\mathcal{R}$ have a common source of randomness which can provide them polynomially as many random bits as they want. Now the shared state variable allows them to pick a new set of random bits and hence a new pseudorandom permutation every time. Hence, as long as the number of queries are polynomial in the security parameter, by the security of pseudorandom permutation we conclude that our claims still hold.