Electronic Colloquium on Computational Complexity, Report No. 19 (2005)

Tolerant Locally Testable Codes

Venkatesan Guruswami[*] Atri Rudra[†]

Department of Computer Science & Engineering
University of Washington
Seattle, WA 98195.
{venkat,atri}@cs.washington.edu

Abstract

An error-correcting code is said to be *locally testable* if it has an efficient spot-checking procedure that can distinguish codewords from strings that are far from every codeword, looking at very few locations of the input in doing so. Locally testable codes (LTCs) have generated a lot of interest over the years, in large part due to their connection to Probabilistically checkable proofs (PCPs). The ability to correct errors that occur during transmission is one of the big advantages of using a code. Hence, from a coding-theoretic angle, local testing is potentially more useful if in addition to accepting codewords, it also accepts strings that are close to a codeword (in contrast, local testers can have arbitrary behavior on such strings, which potentially annuls the benefits of error-correction). This would imply that when the tester accepts, one can follow-up the testing with a (more expensive) decoding procedure to correct the errors and recover the transmitted codeword, while if the tester rejects, we can save the effort of running the more expensive decoding algorithm.

In this work, we define such testers, which we call *tolerant testers* following some recent work in property testing [13]. We revisit some recent constructions of LTCs and show how one can make them locally testable in a tolerant sense. While we do not optimize the parameters, the main message from our work is that there are explicit tolerant LTCs with similar parameters to LTCs.

1 Introduction

Locally testable codes (LTCs) have been the subject of much research over the years and there has been heightened activity and progress on them recently [10, 4, 12, 5, 6, 9]. LTCs are error-correcting codes which have a testing procedure with the following property: given oracle access to a string which is a purported codeword, these testers "spot check" the string at very few locations, accepting if the string is indeed a codeword, and rejecting with high probability if the string is "far-enough" from every codeword. Such spot-checkers arise in the construction of Probabilistically checkable proofs (PCPs) [1, 2] (see the recent survey by Goldreich [9] for more details on the interplay between LTCs and PCPs). Note that in the definition of LTCs, there is no requirement on the tester for

[*]Supported in part by NSF CAREER Award CCF-0343672.
[†]Supported in part on NSF grant CCF-0343672.

ISSN 1433-8092

input strings that are very close to a codeword. This "asymmetry" in the way the tester accepts and rejects an input reflects the way PCPs are defined, where we only care about accepting perfectly correct proofs with high probability. However, the crux of error-correcting codes is to tolerate and *correct* a few errors that could occur during transmission of the codeword (and not just be able to detect errors). In this context, the fact that a tester can reject received words with few errors is not satisfactory. A more desirable (and stronger) requirement in this scenario would be the following: we would like the tester to make a quick decision on whether or not the purported codeword is close to any codeword. If the tester declares that there is probably a close-by codeword, we then use a decoding algorithm to decode the received word. If on the other hand, we can say with high confidence that the received word is far away from all codewords then we do not run our expensive decoding algorithm. The current testers in the literature focus more or less exclusively on the latter goal. As a concrete example of why we should accept received words with few errors, if the received word is at most distance e away from a Reed-Solomon codeword where e is some quantity smaller than half the minimum distance of the Reed-Solomon code, then it would be nice if the tester does not reject the received word as we can uniquely decode it.

In this work we introduce the concept of *tolerant* testers, that is, we design testers for codes which reject (w.h.p) received words far from any codeword (like the current testers) and accept (w.h.p) close-by received words (unlike the current ones which only need to accept codewords). We will refer to codes that admit a tolerant tester as a tolerant LTCs. In the general context of property testing, the notion of tolerant testing was introduced by Parnas *et al* [13] along with the related notion of distance approximation. Parnas *et al* also give tolerant testers for clustering. We feel that codeword-testing is a particularly natural instance to study tolerant testing. (In fact, if LTCs were defined solely from a coding-theoretic viewpoint, without their relevance and applications to PCPs in mind, we feel that it is likely that the original definition itself would have required tolerant testers.)

For any vectors $u, v \in \mathbb{F}_q^n$, the relative distance between u and v, denoted $\text{dist}(u,v)$, equals the fraction of positions where u and v differ. For any subset $A \subset \mathbb{F}_q^n$, $\text{dist}(v,A) = \min_{u\in A}\text{dist}(u,v)$. An $[n,k,d]_q$ linear code $\mathcal{C}$ is a k-dimensional subspace of $\mathbb{F}_q^n$ such that every pair of distinct elements $x, y \in \mathcal{C}$ differ in at least d locations, i.e., $\text{dist}(x,y) \geq d/n$. The ratio $\frac{k}{n}$ is called the rate of the code and d is the (minimum) distance of the code. We now formally define a *tolerant* tester.

Definition 1 *For any linear code $\mathcal{C}$ over $\mathbb{F}_q$ of block length n and distance d, and $0 \leq c_1 \leq c_2 \leq 1$, a (c_1, c_2)-tolerant tester T for $\mathcal{C}$ with query complexity $p(n)$ (or simply p when the argument is clear from the context) is a probabilistic polynomial time[1] oracle Turing machine such that for every vector $v \in \mathbb{F}_q^n$:*

1. *If $\text{dist}(v,\mathcal{C}) \leq \frac{c_1 d}{n}$, T upon oracle access to v accepts with probability at least $\frac{2}{3}$ (*tolerance*),*
2. *If $\text{dist}(v,\mathcal{C}) > \frac{c_2 d}{n}$, T rejects with probability at least $\frac{2}{3}$ (*soundness*),*
3. *T makes $p(n)$ probes into the string (oracle) v.*

A code is said to be (c_1, c_2, p)-testable if it admits a (c_1, c_2)-tolerant tester of query complexity $p(\cdot)$.

[1]In the usual definition of LTCs one omits the requirement that the tester be efficient. This is because the focus usually is on testers which make $O(1)$ queries and in this case an efficient implementation is obvious. We will be interested in testers with sub-linear query complexities that grow with n (like n^γ for small $\gamma > 0$) and therefore we stipulate this requirement. Note that since our results are positive, i.e., we gives constructions of tolerant LTCs, this only makes our results stronger.

We will be interested in asymptotics and thus we implicitly are interested in a family of codes with the stated properties in the above definition (and so, the notion of the tester being a polynomial time machine, in particular, makes sense). We usually hide this for notational simplicity.

A tester has *perfect completeness* if it accepts any codeword with probability 1. As pointed out earlier, the existing literature just consider $(0, c_2)$-tolerant testers with perfect completeness. We will refer to these as *standard* testers henceforth. Note that our definition of tolerant testers is per se not a generalization of standard testers since we do not require perfect completeness for the case when the input v is a codeword. However, all our constructions will inherit this property from the standard testers we obtain them from.

Recall one of the applications of tolerant testers mentioned earlier: a tolerant tester is used to decide if the expensive decoding algorithm should be used. In this scenario, one would like to set the parameters c_1 and c_2 such that the tester is tolerant up to the decoding radius. For example, if we have an unique decoding algorithm which can correct up to $\frac{d}{2}$ errors, a particularly appealing setting of parameters would be $c_1 = \frac{1}{2}$ and c_2 as close to $\frac{1}{2}$ as possible. However, we would not be able to achieve such large c_1. In general we will aim for positive constants c_1 and c_2 with $\frac{c_2}{c_1}$ being as small as possible while minimizing $p(n)$.

One might hope that the existing standard testers could also be tolerant testers. We give a simple example to illustrate the fact that this is not the case in general. Consider the tester for the Reed-Solomon (RS) codes of dimension $k+1$: pick $k+2$ points uniformly at random and check if the degree k univariate polynomial obtained by interpolating on the first $k+1$ points agrees with the input on the last point. It is well known that this is a standard tester [16]. However, this is not a tolerant tester. Assume we have an input which differs from a degree k polynomial in only one point. Thus, for $\binom{n-1}{k+1}$ choices of $k+2$ points, the tester would reject, that is, the rejection probability is $\frac{\binom{n-1}{k+1}}{\binom{n}{k+2}} = \frac{k+2}{n}$ which is greater than $\frac{1}{3}$ for high rate RS codes.

Another pointer towards the inherent difficulty in coming up with a tolerant tester is the recent work of Fischer and Fortnow [7] which shows that there are certain boolean properties which have a standard tester with constant number of queries but for which every tolerant tester requires at least $n^{\Omega(1)}$ queries.

In this work, we examine existing standard testers and convert some standard testers into tolerant ones. In Section 2 we record a few general facts which will be useful in performing this conversion. The ultimate goal, if this can be realized at all, would be to construct tolerant LTCs of constant rate which can be tested using $O(1)$ queries (we remark that such a construction has not been obtained even without the requirement of tolerance). In this work, we show that we can achieve either constant number of queries with slightly sub-constant rate (Section 3) as well as constant rate with sub-linear number of queries (Section 4). That is, something non-trivial is possible in both the domains: (a) constant rate, and (b) constant number of queries. Specifically, in Section 3 we discuss binary codes which encode k bits into codewords of length $n = k \cdot \exp(\log^{\varepsilon} k)$ for any $\varepsilon > 0$, and can be tolerant tested using $O(1/\varepsilon)$ queries. In Section 4, following [5], we study the simple construction of LTCs using products of codes — this yields asymptotically good codes which are tolerant testable using a sub-linear number n^{γ} of queries for any desired $\gamma > 0$. An interesting common feature of the codes in Section 3 and 4 is that they can be constructed from any code that has good distance properties and which in particular need not admit a local tester with sub-linear query complexity. In Section 5 we discuss the tolerant testability of Reed-Muller codes — it turns out that existing results on low-degree testing of multivariate polynomials immediately imply results on tolerant testing for these codes.

The overall message from our work is that a lot of the work on locally testable code constructions extends fairly easily to also yield tolerant locally testable codes. However, there does not seem to be a generic way to "compile" a standard tester to a tolerant tester for an arbitrary code.

2 General Observations

In this section we will fix some notations and spell out some general properties of tolerant testers and subsequently use them to design tolerant testers for some existing codes.

We will denote the set $\{1, \cdots, n\}$ by $[n]$. All the testers we refer to are non-adaptive testers which decide on the locations to query all at once based only on the random choices. In the sequel, we use n to denote the block length and d the distance of the code under consideration. The motivation for the definition below will be clear in Section 3.

Definition 2 *Let* $0 < \alpha \leq 1$. *A tester* T *is* $(\langle s_1, q_1\rangle, \langle s_2, q_2\rangle, \alpha)$-smooth *if there exists a set* $A \subseteq [n]$ *where* $|A| = \alpha n$ *with the following properties:*

- *T queries at most q_1 points in A, and for every $x \in A$, the probability that each of these queries equals location x is at most $\frac{s_1}{|A|}$, and*
- *T queries at most q_2 points in $[n] - A$, and for every $x \in [n] - A$, the probability that each of these queries equals location x is at most $\frac{s_2}{n-|A|}$.*

As a special case a $(\langle 1, q\rangle, \langle 0, 0\rangle, 1)$-smooth tester makes a total of q queries each of them distributed uniformly among the n possible probe points.

The following lemma follows from the union bound:

Lemma 1 *For any* $0 < \alpha < 1$, *a* $(\langle s_1, q_1\rangle, \langle s_2, q_2\rangle, \alpha)$*-smooth* $(0, c_2)$*-tolerant tester* T *with perfect completeness is a* (c_1, c_2)*-tolerant tester* T', *where* $c_1 = \frac{n\alpha(1-\alpha)}{3d\max\{q_1 s_1(1-\alpha),\ q_2 s_2 \alpha\}}$.

Proof: The soundness follows from the assumption on T. Assume $\mathrm{dist}(v, \mathcal{C}) \leq \frac{c_1 d}{n}$ and let $f \in \mathcal{C}$ be the closest codeword to v. Suppose that f differs from v in a set A' of yd places among locations in A, and a set B' of $(\beta - y)d$ places among locations in $[n] - A$, where we have $\beta \leq c_1$ and $0 \leq y \leq \beta$. The probability that any of the at most q_1 (resp. q_2) queries of T into A (resp. $[n] - A$) falls in A' (resp. B') is at most $\frac{s_1 y d}{\alpha n}$ (resp. $\frac{s_2(\beta - y)d}{(1-\alpha)n}$). Clearly, whenever T does not query a location in $A' \cup B'$, it accepts (since T has perfect completeness). Thus, an easy calculation shows that the probability that T rejects v is at most

$$\frac{c_1 d}{n} \max\{\frac{s_1 q_1}{\alpha}, \frac{s_2 q_2}{1-\alpha}\}$$

which is 1/3 for the choice of c_1 stated in the lemma. ■

The above lemma is not satisfactory unless the relative distance and the number of queries are constants. Next we sketch how to design tolerant testers from existing *robust* testers with certain properties. We first recall the definition of robust testers from [5].

A standard tester T has two inputs: an oracle for the received word v and a random string s. Depending on s, T generates q query positions $i_1, \cdots, i_q$, fixes a circuit C_s and then accepts if $C_s(v_f(s)) = 1$ where $v_f(s) = \langle v_{i_1}, \cdots, v_{i_q} \rangle$. The robustness of T on inputs v and s, denoted by $\rho^T(v, s)$, is defined to be the minimum, over all strings y such that $C_s(y) = 1$, of $\text{dist}(v_f(s), y)$. The expected robustness of T on v is the expected value of $\rho^T(v, s)$ over the random choices of s and would be denoted by $\rho^T(v)$.

A standard tester T is said to be c-robust for $\mathcal{C}$ if for every $v \in \mathcal{C}$, the tester accepts with probability 1, and for every $v \in \mathbb{F}_q^n$, $\text{dist}(v, \mathcal{C}) \leq c \cdot \rho^T(v)$.

The tolerant version T' of the standard c-robust tester T is obtained by accepting an oracle v on random input s, if $\rho^T(v, s) \leq \tau$ for some threshold τ. (Throughout the paper τ will denote the threshold.) We will sometimes refer to such a tester as one with threshold τ. Recall that a standard tester T accepts if $\rho^T(v, s) = 0$. We next show that T' is sound. For the rest of this section unless mentioned otherwise, we will use parameter τ to denote the threshold.

The following lemma follows from the fact that T is c-robust:

Lemma 2 *Let $0 \leq \tau \leq 1$, and let $c_2 = \frac{(\tau+2)cn}{3d}$. For any $v \in \mathbb{F}_q^n$, if $\text{dist}(v, \mathcal{C}) > \frac{c_2 d}{n}$, then the tolerant tester T' with threshold τ rejects v with probability at least $\frac{2}{3}$.*

Proof: Let $v \in \mathbb{F}_q^n$ be such that $\text{dist}(v, \mathcal{C}) > \frac{c_2 d}{n}$. By the definition of robustness, the expected robustness, $\rho^T(v)$ is at least $\frac{c_2 d}{nc}$, and thus at least $(\tau + 2)/3$ by the choice of c_2. By the standard averaging argument, we can have $\rho^T(v, s) \leq \tau$ on at most a fraction 1/3 of the of the random choices of s for T (and hence T'). Therefore, $\rho^T(v, s) > \tau$ with probability at least 2/3 over the choice of s and thus T' rejects v with probability at least 2/3. ■

We next mention a property of the query pattern of T which would make T' tolerant. Let S be the set of all possible choices for the random string s. Further for each s, let $p^T(s)$ be the set of positions queried by T.

Definition 3 *A tester T has a* partitioned *query pattern if there exists a partition $S_1 \cup \cdots \cup S_m$ of the random choices of T for some m, such that for every i,*

- *$\cup_{s \in S_i} p^T(s) = \{1, 2, \cdots, n\}$, and*
- *For all $s, s' \in S_i$, $p^T(s) \cap p^T(s') = \emptyset$ if $s \neq s'$.*

Lemma 3 *Let T have a partitioned query pattern. For any $v \in \mathbb{F}_q^n$, if $\text{dist}(v, \mathcal{C}) \leq \frac{c_1 d}{n}$, where $c_1 = \frac{n\tau}{3d}$, then the tolerant test T' with threshold τ rejects with probability at most $\frac{1}{3}$.*

Proof: Let $S_1, \cdots, S_m$ be the partition of S, the set of all random choices of the tester T. For each j, by the properties of S_j, $\sum_{s \in S_j} \rho^T(v, s) \leq \text{dist}(v, \mathcal{C})$. By an averaging argument and by the assumption on $\text{dist}(v, f)$ and the value of c_1, at least $\frac{2}{3}$ fraction of the choices of s in S_j have $\rho^T(v, s) \leq \tau$ and thus, T' accepts. Recalling that $S_1, \cdots, S_m$ was a partition of S, for at least $\frac{2}{3}$ of the choices of s in S, T' accepts. This completes the proof. ■

3 Tolerant Testers for Binary Codes

One of the natural goals in the study of tolerant codes is to design explicit tolerant binary codes with constant relative distance and as large a rate as possible. In the case of standard testers, Ben-Sasson et al [4] give binary locally testable codes which map k bits to $k \cdot \exp(\log^{\varepsilon} k)$ bits for any $\varepsilon > 0$ and which are testable with $O(1/\varepsilon)$ queries. Their construction uses objects called PCPs of Proximity (PCPP) which they also introduce in [4]. In this section, we show that a simple modification to their construction yields tolerant testable binary codes which map k bits to $k \cdot \exp(\log^{\varepsilon} k)$ bits for any $\varepsilon > 0$. We note that a similar modification is used by Ben-Sasson et al to give a relaxed locally decodable codes [4] but with worse parameters (specifically they gives codes with block length $k^{1+\varepsilon}$).

3.1 PCP of Proximity

We start with the definition[2] of of a Probabilistic Checkable proof of Proximity (PCPP). A pair language is simply a language whose elements are naturally a pair of strings, i.e., it is some collection of strings (x, y). A notable example is CIRCUITVAL $= \{\langle C, a\rangle \mid$ Boolean circuit C evaluates to 1 on assignment $a\}$.

Definition 4 *Fix $0 \le \delta \le 1$. A probabilistic verifier V is a PCPP for a pair language L with proximity parameter δ and query complexity $q(\cdot)$ if the following conditions hold:*

- *(Completeness) If $(x, y) \in L$ then there exists a proof π such that V accepts by accessing the oracle $y \circ \pi$ with probability 1.*
- *(Soundness) If y is δ-far from $L(x) = \{y|(x, y) \in L\}$, then for all proofs π, V accepts by accessing the oracle $y \circ \pi$ with probability strictly less than $\frac{1}{4}$.*
- *(Query complexity) For any input x and proof π, V makes at most $q(|x|)$ queries in $y \circ \pi$.*

Note that a PCPP differs from a standard PCP in that it has a more relaxed soundness condition but its queries into part of the input y are also counted in its query complexity.

Ben-Sasson et. al. give constructions of PCPPs with the following guarantees:

Lemma 4 *([4]) Let $\varepsilon > 0$ be arbitrary. There exists a PCP of proximity for the pair language* CIRCUITVAL $= \{(C, x)|C$ *is a boolean circuit and* $C(x) = 1\}$ *whose proof length, for inputs circuits of size s, is at most $s \cdot \exp(\log^{\varepsilon/2} s)$ and for $t = \frac{2 \log \log s}{\log \log \log s}$ the verifier of proximity has query complexity $O(\max\{\frac{1}{\delta}, \frac{1}{\varepsilon}\})$ for any proximity parameter δ that satisfies $\delta \ge \frac{1}{t}$. Furthermore, the queries of the verifier are non-adaptive and each of the queries which lie in the input part x are uniformly distributed among the locations of x.*

The fact that the queries to the input part are uniformly distributed follows by an examination of the verifier construction in [4]. In fact, in the extended version of that paper, the authors make this fact explicit and use it in their construction of relaxed locally decodable codes (LDCs). To

[2]The definition here is a special case of the general PCPP defined in [4] which would be sufficient for our purposes.

achieve a tolerant LTC using the PCPP, we will need all queries of the verifier to be somewhat uniformly or smoothly distributed. We will now proceed to make the queries of the PCPP verifier that fall into the "proof part" π near-uniform. This will follow a fairly general method suggested in [4] to smoothen out the query distribution, which the authors used to obtain relaxed locally decodable codes from the PCPP. We will obtain tolerant LTCs instead, and in fact will manage to do so without a substantial increase in the encoding length (i.e., the encoding length will remain $k \cdot 2^{\log^{\varepsilon} k}$). On the other hand, the best encoding length achieved for relaxed LDCs in [4] is $k^{1+\varepsilon}$ for constant $\varepsilon > 0$. We begin with the definition of a mapping that helps smoothen out the query distribution.

Definition 5 *Given any $v \in \mathbb{F}_q^n$ and $\vec{p} = \langle p_i \rangle_{i=1}^n$ with $p_i \geq 0$ for all $i \in [n]$ and $\sum_{i=1}^n p_i = 1$, we define the mapping $\mathsf{Repeat}(\cdot,\cdot)$ as follows: $\mathsf{Repeat}(v,\vec{p}) \in \mathbb{F}_q^{n'}$ such that v_i is repeated $\lfloor 4np_i \rfloor$ times in $\mathsf{Repeat}(v,\vec{p})$ and $n' = \sum_{i=1}^n \lfloor 4np_i \rfloor$.*

We now show why the mapping is useful. A similar fact appears in [4], but for the sake of completeness we present the proof.

Lemma 5 *For any $v \in \mathbb{F}_q^n$ let a non-adaptive verifier T (with oracle access to v) make $q(n)$ queries and let p_i be the probability that each of these queries probes location $i \in [n]$. Let $c_i = \frac{1}{2n} + \frac{p_i}{2}$ and $\vec{c} = \langle c_i \rangle_{i=1}^n$. Consider the map $\mathsf{Repeat}(v,\vec{c}) : \mathbb{F}_q^n \to \mathbb{F}_q^{n'}$. Then there exists another tester T' for strings of length n' with the following properties:*

1. *T' makes $2q(n)$ queries on $v' = \mathsf{Repeat}(v,\vec{c})$ each of which probes location j, for any $j \in [n']$, with probability at most $\frac{2}{n'}$, and*
2. *for every $v \in \mathbb{F}_q^n$, the decision of T' on v' is identical to that of T on v. Further, $3n < n' \leq 4n$.*

Proof: We first add q dummy queries to T each of which are uniformly distributed, and then permute the $2q$ queries in a random order. Note that each of the $2q$ queries is now identically distributed. Moreover, any position in v is probed with probability at least $\frac{1}{2n}$ for each of the $2q$ queries. For the rest of the proof we will assume that T makes $2q$ queries for each of which any $i \in [n]$ is probed with probability $c_i = \frac{p_i}{2} + \frac{1}{2n}$. Let $r_i = \lfloor 4nc_i \rfloor$. Note that $r_i \leq 4nc_i$ and $r_i > 4nc_i - 1$. Recalling that $n' = \sum_{i=1}^n r_i$ and $\sum_{i=1}^n c_i = 1$, we have $3n < n' \leq 4n$.

T' just simulates T in the following manner: if T queries v_i for any $i \in [n]$, T' queries one of the r_i copies of v_i in v' uniformly at random. It is clear that the decision of T' on $v' = \mathsf{Repeat}(v,\vec{c})$ is identical to that of T on v. We now look at the query distribution of T'. T' queries any $j \in [n']$, where $v'_j = v_i$, with probability $p'_j = c_i \cdot \frac{1}{r_i}$. Recalling the lower bound on r_i, we have $p'_j \leq \frac{c_i}{4nc_i - 1}$ which is at most $\frac{1}{2n}$ since clearly $c_i \geq \frac{1}{2n}$. We showed earlier that $n' \leq 4n$ which implies $p'_j \leq \frac{2}{n'}$ as required. ■

One might wonder if we can use Lemma 5 to smoothen out the queries made by the verifier of an arbitrary LTC to obtain a tolerant LTC. That is, whether the above allows one to compile the verifier for any LTC in a black-box manner to obtain a tolerant verifier. This does not seem likely, and we discuss this further in subsection 3.3.

Applying the transformation of Lemma 5 to the proximity verifier and proof of proximity of Lemma 4, we conclude the following.

Proposition 1 *Let $\varepsilon > 0$ be arbitrary. There exists a PCP of proximity for the pair language* CIRCUITVAL $= \{(C, x) | C$ *is a boolean circuit and* $C(x) = 1\}$ *with the following properties:*

1. *The proof length, for inputs circuits of size s, is at most $s \cdot \exp(\log^{\varepsilon/2} s)$, and*
2. *for $t = \frac{2 \log \log s}{\log \log \log s}$ the verifier of proximity has query complexity $O(\max\{\frac{1}{\delta}, \frac{1}{\varepsilon}\})$ for any proximity parameter δ that satisfies $\delta \geq \frac{1}{t}$.*

Furthermore, the queries of the verifier are non-adaptive with the following properties:

1. *Each query made to one of the locations of the input x is uniformly distributed among the locations of x, and*
2. *each query to one of the locations in the proof of proximity π probes each location with probability at most $2/|\pi|$ (and thus is distributed* nearly uniformly *among the locations of π).*

3.2 The Code

We now outline the construction of the locally testable code from [4]. The idea behind the construction is to make use of a PCPP to aid in checking if the received word is a codeword is far away from being one. Details follow.

Suppose we have a binary code $C_0 : \{0,1\}^k \to \{0,1\}^m$ of distance d defined by a parity check matrix $H \in \{0,1\}^{(m-k)\times m}$ that is sparse, i.e., each of whose rows has only an absolute constant number of 1's. Such a code is referred to as a low-density parity check code (LDPC). For the construction below, we will use any such code which is asymptotically good (i.e., has rate k/m and relative distance d/m both positive as $m \to \infty$). Explicit constructions of such codes are known using expander graphs [15]. Let V be a verifier of a PCP of proximity for membership in C_0; more precisely, the proof of proximity of an input string $w \in \{0,1\}^m$ will be a proof that $\tilde{C_0}(w) = 1$ where $\tilde{C_0}$ is a linear-sized circuit which performs the parity checks required by H on w (the circuit will have size $O(m) = O(k)$ since H is sparse and C_0 has positive rate). Denote by $\pi(x)$ be the proof of proximity guaranteed by Proposition 1 for the claim that the input $C_0(x)$ is a member of C_0 (i.e., satisfies the circuit $\tilde{C_0}$). By Proposition 1 and fact that the size of $\tilde{C_0}$ is $O(k)$, the length of $\pi(x)$ can be made at most $k \exp(\log^{\varepsilon/2} k)$.

The final code is defined as $\mathcal{C}_1(x) = (C_0(x)^t, \pi(x))$ where $t = \frac{(\log k - 1)|\pi(x)|}{|C_0(x)|}$. The repetition of the code part $C_0(x)$ is required in order to ensure good distance, since the length of the proof part $\pi(x)$ typically dominates and we have no guarantee on how far apart $\pi(x_1)$ and $\pi(x_2)$ for $x_1 \neq x_2$ are.

For the rest of this section let ℓ denote the proof length. The tester T_1 for $\mathcal{C}_1$ on an input $w = (w_1, \cdots, w_t, \pi) \in \{0,1\}^{tm+\ell}$ picks $i \in [t]$ at random and runs the PCPP verifier V on $w_i \circ \pi$. It also performs a few rounds of the following consistency checks: pick $i_1, i_2 \in [t]$ and $j_1, j_2 \in [m]$ at random and check if $w_{i_1}(j_1) = w_{i_2}(j_2)$. Ben-Sasson et al in [4] show that T_1 is a standard tester. However, T_1 need not be a tolerant tester. To see this, note that the proof part of $\mathcal{C}_1$ forms a $\frac{1}{\log k}$ fraction of the total length. Now consider a received word $w_{rec} = (w_0, \cdots, w_0, \pi')$ where $w_0 \in C_0$ but π' is not a correct proof for w_0 being a valid codeword in c_0. Note that w_{rec} is close to $\mathcal{C}_1$. However, T_1 is not guaranteed to accept w_{rec} with high probability.

The problem with the construction above was that the proof part was too small: a natural fix is to make the proof part a constant fraction of the codeword. We will show that this is sufficient to make the code tolerant testable. We also remark that a similar idea was used by Ben-Sasson et. al. to give efficient constructions for relaxed locally decodable codes [4].

Construction 1 *Let $0 < \beta < 1$ be a parameter, $C_0 : \{0,1\}^k \to \{0,1\}^m$ be a good*[3] *binary code and V be a PCP of proximity verifier for membership in C_0. Finally let $\pi(x)$ be the proof corresponding to the claim that $C_0(x)$ is a codeword in C_0. The final code is defined as $\mathcal{C}_2(x) = (C_0(x)^{r_1}, \pi(x)^{r_2})$ with $r_1 = \frac{(1-\beta)\log k|\pi(x)|}{m}$ and $r_2 = \beta \log k$.*[4]

For the rest of the section the proof length $|\pi(x)|$ will be denoted by ℓ. Further the proximity parameter and the number of queries made by the PCPP verifier V would be denoted by δ_p and q_p respectively. Finally let ρ_0 denote the relative distance of the code C_0.

The tester T_2 for $\mathcal{C}_2$ is also the natural generalization of T_1. For a parameter q_r (to be instantiated later) and input $w = (w_1, \cdots, w_{r_1}, \pi_1, \cdots, \pi_{r_2}) \in \{0,1\}^{r_1 m + r_2 l}$, T_2 does the following:

1. Repeat the next two steps twice.
2. Pick $i \in [r_1]$ and $j \in [r_2]$ randomly and run V on $w_i \circ \pi_j$.
3. Do q_r repetitions of the following: pick $i_1, i_2 \in [r_1]$ and $j_1, j_2 \in [m]$ randomly and check if $w_{i_1}(j_1) = w_{i_2}(j_2)$.

The following lemma captures the properties of the code $\mathcal{C}_2$ and its tester T_2.

Lemma 6 *The code $\mathcal{C}_2$ in Construction 1 and the tester T_2 (with parameters β and q_r respectively) above have the following properties:*

1. *The code $\mathcal{C}_2$ has block length $n = \log k \cdot \ell$ with minimum distance d lower bounded by $(1-\beta)\rho_0 n$.*
2. *T_2 makes a total of $q = 2q_p + 4q_r$ queries.*
3. *T_2 is $(\langle 1, q\rangle, \langle 2, 2q_p\rangle, 1-\beta)$-smooth.*
4. *T_2 is a (c_1, c_2)-tolerant tester with $c_1 = \frac{n\beta(1-\beta)}{6d\max\{(2q_r+q_p)\beta,\ 2(1-\beta)q_p\}}$ and $c_2 = \frac{n}{d}(\delta_p + \frac{4}{q_r} + \beta)$.*

Proof: From the definition of $\mathcal{C}_2$, it has block length $n = r_1 m + r_2 \ell = \frac{(1-\beta)\ell\log k}{m} \cdot m + \beta \log k \cdot \ell = \log k \cdot \ell$. Further as C_0 has relative distance ρ_0, $\mathcal{C}_2$ has relative distance at least $\frac{r_1 \rho_0 m}{\ell \log k} = (1-\beta)\rho_0$.

T_2 makes the same number of queries as V which is q_p in Step 2. In Step 3, T_2 makes $2q_r$ queries. As T_2 repeats Steps 2 and 3 twice, we get the desired query complexity.

To show the smoothness of T_2 we need to define the appropriate subset $A \subset [n]$ such that $|A| = (1-\beta)n$. Let A be the set of indices with the code part: i.e. $A = [r_1 m]$. T_2 makes $2q_r$ queries in A

[3]This means that $m = O(k)$ and the encoding can be done by circuits of nearly linear size $s_0 = \tilde{O}(k)$.

[4]The factor $\log k$ overhead is overkill, and a suitably large constant will do, but since the proof length $|\pi(x)|$ will anyway be larger than $|x|$ by more than a polylogarithmic factor in the constructions we use, we can afford this additional $\log k$ factor and this eases the presentation somewhat.

in Step 3 each of which is uniformly distributed. Further by Proposition 1, T_2 in step 2 makes at most q_p queries in A which are uniformly distributed and at most q_p queries in $[n]-A$ each of which are within a factor 2 of being queried uniformly at random. To complete the proof of property 3 note that T_2 repeats step 2 and 3 twice.

The tolerance of T_2 follows from property 3 and Lemma 1. For the soundness part note that if $w = (w_1, \cdots, w_{r_1}, \pi_1, \cdots, \pi_{r_2}) \in \{0,1\}^{r_1 m + r_2 l}$ is δ-far from $\mathcal{C}_2$ then $(w_1, \cdots, w_{r_1})$ is at least $\frac{\delta n - r_2\ell}{n} = \frac{\delta n - \beta n}{n} = \delta - \beta$ far from the repetition code $C' = \{C_0(x)^{r_1} | x \in \{0,1\}^k\}$. For $\delta = c_2 d/n$ with the choice of c_2 in the lemma, we have $\delta - \beta \geq \delta_p + 4/q_r$. The rest of the proof just follows the proof in [4] of the soundness of the tester T_1 for the code $\mathcal{C}_1-$ as in [4] one can show that one invocation of Steps 2 and 3 results in T_2 accepting w with probability strictly less than $\frac{1}{2}$. The two repetitions of Steps 2 and 3 reduces this error to at most $\frac{1}{4}$. ■

Fix any $0 < \delta < 1$ and let $\beta = \frac{\delta}{2}$, $\delta_p = \frac{\delta}{6}$, $q_r = \frac{12}{\delta}$. With these settings we get $\delta_p + \frac{4}{q_r} + \beta = \delta$ and $q_p = O(\frac{1}{\delta})$ from Proposition 1 with the choice $\varepsilon = 2\delta$. Finally, $q = 2q_p + 4q_r = O(\frac{1}{\delta})$. Substituting the parameters in c_2 and c_1, we get $c_2 = \frac{\delta n}{d}$ and

$$\frac{c_1 d}{n} = \frac{\delta}{24 \max\{\delta(q_r + q_p/2),\ (2-\delta)q_p\}} = \Omega(\delta^2)\ .$$

Also note that the minimum distance $d \geq (1-\beta)\rho_0 n = (1 - \frac{\delta}{2})\rho_0 n \geq \frac{\rho_0}{2} n$. Thus, we have the following result for tolerant testable binary codes.

Theorem 1 *There exists an absolute constant $\alpha_0 > 0$ such that for every δ, $0 < \delta < 1$, there exists an explicit binary linear code $\mathcal{C} : \{0,1\}^k \to \{0,1\}^n$ where $n = k \cdot \exp(\log^\delta k)$ with minimum distance $d \geq \alpha_0 n$ which admits a (c_1, c_2)-tolerant tester with $c_2 = O(\delta)$, $c_1 = \Omega(\delta^2)$ and query complexity $O(\frac{1}{\delta})$.*

The claim about explicitness follows from the fact that the PCPP of Lemma 4 and hence Proposition 1 has an explicit construction. The claim about linearity follows from the fact that the PCPP for CIRCUITVAL is a linear function of the input when the circuit computes linear functions — this aspect of the construction is discussed in detail in Section 8.4 of the extended version of [4].

3.3 Converting locally testable codes to tolerant testable codes

We now discuss whether the technique of repeating symbols in proportion to their query probability as in the previous section (specifically, Lemma 5) can be used to convert a LTC into a tolerant LTC (without, for example, the additional complexity of using a PCPP). We will now argue (informally) that this technique alone will not work. Let C_1 be an $[n,k,d]_q$ LTC with a standard tester T_1 that makes q identically distributed queries with distribution p_i, $1 \leq i \leq n$, such that $p_i \geq 1/2n$ for each i. Create a new $[n+1,k,d]_q$ code C_2 whose $(n+1)$'th coordinate is just a copy of the n'th coordinate, i.e., corresponding to each codeword $(c_1, c_2, \ldots, c_n) \in \mathbb{F}_q^n$ of C_1, we will have a codeword $(c_1, c_2, \ldots, c_n, c_n) \in \mathbb{F}_q^{n+1}$ of C_2. Consider the following tester T_2 for C_2: Given oracle access to $v \in \mathbb{F}_q^{n+1}$, with probability 1/2 check whether $v_n = v_{n+1}$, and with probability 1/2 run the tester T_1 on the first n coordinates of v. Clearly, T_2 is a standard tester for C_2.

Now, consider what happens in the conversion procedure of Lemma 5 to get (C', T') from (C_2, T_2). Note that by Lemmas 5 and 3, T' is tolerant. Let $\vec{q} = (q_1, \ldots, q_{n+1})$ be the query distribution of T_2. Since T_2 queries (v_n, v_{n+1}) with probability $1/2$, the combined number of locations of $v' = \mathsf{Repeat}(v, \vec{q})$ corresponding to v_n, v_{n+1} will be about $1/2$ of the total length n'. Now let v' be obtained from a codeword of C' by corrupting just these locations. The tester T' will accept such a v' with probability at least $1/2$, which contradicts the soundness requirement since v' is $1/2$-far from C'. Therefore, using the behavior of the original tester T_2 as just a black-box, we cannot in general argue that the construction of Lemma 5 maintains good soundness.

4 Tolerant Testers for Tensor Products of Codes

Tensor product of codes is simple way to construct new codes from any existing codes such that the constructed codes have testers with sub-linear query complexity even though the original code need not admit a sub-linear complexity tester [5]. We first briefly define product of codes and then outline the tester of product of codes from [5].

Given an $[n, k, d]_q$ code $\mathcal{C}$, the product of $\mathcal{C}$ with itself, denoted by $\mathcal{C}^2$, is a $[n^2, k^2, d^2]_q$ code such that a codeword (viewed as a $n \times n$ matrix) restricted to any row or column is a codeword in $\mathcal{C}$. More formally, given the $n \times k$ generator matrix M of $\mathcal{C}$, $\mathcal{C}^2$ is precisely the set of matrices in the set $\{M \cdot X \cdot M^T \mid X \in F_q^{k \times k}\}$. A very natural test for $\mathcal{C}^2$ is to randomly choose a row or a column and then check if the restriction of the received word on that row or column is a codeword in $\mathcal{C}$ (which can be done for example by querying all the n points in the row or column). Unfortunately, it is not known if this test is robust in general (see the discussion in [5]).

Ben-Sasson and Sudan in [5] considered the more general product of codes $\mathcal{C}^t$ for $t \geq 3$ along with the following general tester: Choose at random $b \in \{1, \cdots, t\}$ and $i \in \{1, \cdots, n\}$ and check if b^{th} coordinate of the received word (which is an element of $\mathbb{F}_q^{n^t}$) when restricted[5] to i is a codeword in $\mathcal{C}^{t-1}$. It is shown in [5] that this test is robust, in that if a received word is far from $\mathcal{C}^t$, then many of the tested substrings will be far from $\mathcal{C}^{t-1}$. This tester lends itself to recursion: the test for $\mathcal{C}^{t-1}$ can be reduced to a test for $\mathcal{C}^{t-2}$ and so on till we need to check whether a word in $\mathbb{F}_q^{n^2}$ is a codeword of $\mathcal{C}^2$. This last check can done by querying all the n^2 points, out of the n^t points in the original received word, thus leading to a sub-linear query complexity. As shown in [5], the reduction can be done in $\log t$ stages by the standard halving technique.

We now give a tolerant version of the test for product of codes given by Ben-Sasson and Sudan [5]. In what follows $t \geq 4$ will be a power of two. As mentioned above the tester T for the tensor product $\mathcal{C}^t$ reduces the test to checking if some restriction of the given string belong to $\mathcal{C}^2$. For the rest of this section, with a slight abuse of notation let $v_f \in \mathbb{F}_q^{n^2}$ denote the final restriction being tested. In what follows we assume that by looking at all points in any $v \in \mathbb{F}_q^{n^2}$ one can determine if $\text{dist}(v, \mathcal{C}^2) \leq \tau$ in time polynomial in n^2.

The tolerant version of the test of [5] is a simple modification as mentioned in Section 2: reduce the test on $\mathcal{C}^t$ to $\mathcal{C}^2$ as in [5] and then accept if v_f is τ-close to $\mathcal{C}^2$.

First we make the following observation about the test in [5]. The test recurses $\log t$ times to reduce the test to $\mathcal{C}^2$. At step l , the test chooses an random coordinate b_l (this will just be a random bit) and fixes the value of the b_l^{th} coordinate of the current $\mathcal{C}^{\frac{t}{2^l}}$ to an index i_l (where i_l takes values in the range $1 \leq i_l \leq n^{t/2^l}$). The key observation here is that for each fixed choice of $b_1, \cdots, b_{\log t}$,

[5]For the $t = 2$ case b signifies either row or column and i denotes the row/column index.

distinct choices of $i_1, \cdots, i_{\log t}$ correspond to querying disjoint sets n^2 points in the original $v \in \mathbb{F}_q^{n^t}$ string, which together form a partition of all coordinates of v. In other words, T has a *partitioned* query pattern, which will be useful to argue tolerance. For soundness, we use the results in [5], which show that their tester is $C^{\log t}$-robust for $C = 2^{32}$.

Thus, from Lemmas 2 and 3 we have the following result

Theorem 2 *Let $t \geq 4$ be a power of two and $0 \leq \tau \leq 1$. There exist $0 < c_1 < c_2 \leq 1$ with $\frac{c_2}{c_1} = C^{\log t}(1+2/\tau)$ such that the proposed tolerant tester for $\mathcal{C}^t$ is a (c_1, c_2)-tolerant tester with query complexity $N^{2/t}$ where N is the block length of $\mathcal{C}^t$. Further, c_1 and c_2 are constants (independent of N) if t is a constant and $\mathcal{C}$ has constant relative distance.*

Thus, Theorem 2 achieves the goal of a simple construction of tolerant testable codes with sub-linear query complexity, as the following corollary records:

Corollary 1 *For every $\gamma > 0$, there is an explicit family of asymptotically good binary linear codes which are tolerant testable using n^γ queries, where n is the block length of the concerned code. (The rate, relative distance and thresholds c_1, c_2 for the tolerant testing depend on γ.)*

5 Tolerant testing of Reed-Muller codes

In this section, we discuss testers for codes based on multivariate polynomials.

5.1 Bivariate polynomial codes

As mentioned in [5], there are no robust standard testers known for $\mathcal{C}^2$ in general. In this subsection, we consider a special case when $\mathcal{C} = \mathrm{RS}[n, k+1, d = n-k]_q$, that is, the Reed–Solomon code based on evaluation of degree k polynomials over $\mathbb{F}_q$ at n distinct points in the field. We show that the tester for $\mathcal{C}^2$ proposed in [5] is tolerant for this special case. It is well-known (see, for example, Proposition 2 in [14]) that in this case $\mathcal{C}^2$ is the code with codewords being the evaluations of bivariate polynomials over $\mathbb{F}_q$ of degree k in each variable. The problem of low-degree testing for bivariate polynomials is a well-studied one: in particular we use the work of Polishchuk and Spielman [14] who analyze a tester using axis parallel lines. Call a bivariate polynomial to be one of degree (k_1, k_2) if the maximum degrees of the two variables are k_1 and k_2 respectively. In what follows, we denote by $Q' \in \mathbb{F}_q^{n \times n}$ be the received word to be tested (thought of as an $n \times n$ matrix), and let $Q(x, y)$ be the degree (k, k) polynomial whose encoding is closest to Q'.

We now specify the tolerant tester T'. The upper bound of $1 - \sqrt{1 - d/n}$ on τ comes from the fact that this is largest radius for which decoding an $\mathrm{RS}[n, k+1, d]$ code is known to be solvable in polynomial time [11].

1. Fix τ where $0 \leq \tau \leq 1 - \sqrt{1 - d/n}$.
2. With probability $\frac{1}{2}$ choose $b = 0$ or $b = 1$.
    - If $b = 0$, choose a row r randomly and reject if $\mathrm{dist}(Q'(r, \cdot), P(\cdot)) > \tau$ for every univariate polynomial P of degree k and accept otherwise.

- If $b = 1$, choose a column c randomly and reject if $\mathrm{dist}(Q'(\cdot, c), P(\cdot)) > \tau$ for every univariate polynomial P of degree k and accept otherwise.

To analyze T' let $R^*(r, \cdot)$ be the closest degree k univariate polynomial (breaking ties arbitrarily) for each row r. Similarly construct $C^*(\cdot, c)$. We will use the following refinement of the Bivariate testing lemma of [14]:

Lemma 7 *([14, 6]) There exists an universal constant $c_0 \leq 128$ such that the following holds. If $8k \leq n$ then*

$$\mathrm{dist}(Q', \mathcal{C}^2) = \mathrm{dist}(Q', Q) \leq c_0 \cdot (\mathrm{dist}(R^*, Q') + \mathrm{dist}(C^*, Q'))$$

The following proposition shows that the standard tester version of T' (that is T' with $\tau = 0$) is a robust tester:

Proposition 1 *T' with $\tau = 0$ is a $2c_0$ robust tester, where c_0 is the constant from Lemma 7.*

Proof: By the definition of the row polynomial R, for any row index r, the robustness of the tester with $b = 0$ and r, $\rho(Q', \langle b, r\rangle) = \mathrm{dist}(Q'(r, \cdot), R^*(r, \cdot))$. Similarly for $b = 1$, we have $\rho(Q', \langle b, c\rangle) = \mathrm{dist}(Q'(\cdot, c), C^*(\cdot, c))$. Now the expected robustness of the test is given by

$$\begin{aligned}\rho(Q') &= \Pr[b = 0] \sum_{i=1}^{n} \Pr[r = i] \cdot \mathrm{dist}(Q'(r, \cdot), R^*(r, \cdot)) + \Pr[b = 1] \sum_{j=1}^{n} \Pr[c = j] \cdot \mathrm{dist}(Q'(\cdot, c), C^*(\cdot, c)) \\ &= \frac{1}{2}(\mathrm{dist}(Q', R^*) + \mathrm{dist}(Q', C^*)) \ .\end{aligned}$$

Using Lemma 7, we get $\mathrm{dist}(Q', Q) \leq 2c_0\rho(Q')$, as required. ■

From the description of T', it is clear that it has a *partitioned* query pattern. There are two partitions: one for the rows (corresponding to the choice $b = 0$) and one for the columns (corresponding to the choice $b = 1$).

Thus, Lemmas 2 and 3 show that T' is a tolerant tester:

Theorem 3 *Let c_0 being the constant from Lemma 7. For $\tau \leq 1 - \sqrt{1 - d/n}$, the tester T' with threshold τ is a $(c_1, c_2, \sqrt{N})$-tolerant tester for $\mathcal{C}^2$ (where $\mathcal{C} = \mathrm{RS}[n, k+1, d]$) where $c_1 = \frac{n\tau}{3d}$, $c_2 = \frac{2nc_0(\tau+2)}{3d}$ and N is the block length of $\mathcal{C}^2$.*

5.2 General Reed-Muller codes

We now turn our attention to testing of general Reed-Muller codes. That is, given a function $f : \mathbb{F}_q^m \to \mathbb{F}_q$ as a table of values, one has to test if f is (close to) a m-variate polynomial of total degree k. (The results of the previous section were for polynomials which had degree in each *individual* variable bounded by some value; here we study the total degree case.) Let us denote by $\mathcal{RM}(k, m, q)$ the linear code consisting of evaluations of degree k m-variate polynomials at all points in $\mathbb{F}_q^m$. Our starting point is the following natural and by now well-studied test which we call the *lines test* (its analysis appears among other places in [8]): pick a random line in $\mathbb{F}_q^m$ and check if the restriction of f on the line is a univariate polynomial of degree at most k. For any $x, h \in \mathbb{F}_q^m$, a line passing through x in direction h is given by the set $L_{x,h} = \{x + th | t \in \mathbb{F}_q\}$. Further define $P_{x,h}^f(\cdot)$ to be the univariate polynomial of degree at most k which is closest (in Hamming distance) from the restriction of f on $L_{x,h}$. We will use the following result.

Theorem 4 *([8]) There exists a constant c such that for all k, if q is a prime power that is at least ck, then given a function $f : \mathbb{F}_q^m \to \mathbb{F}_q$ with*

$$\rho \overset{\text{def}}{=} E_{x,h \in \mathbb{F}_q^m} \Pr_{t \in \mathbb{F}_q}[P_{x,h}^f(t) \neq f(x + th)] \leq \frac{1}{9} \ ,$$

there exists an m-variate polynomial g of total degree at most k such that $dist(f, g) \leq 2\rho$.

The above result clearly implies that the line test is robust which we record in the following corollary.

Corollary 2 *There exists a constant c such that the line test for $\mathcal{RM}(k, m, q)$ with $q \geq ck$ is 9-robust.*

The line test picks a random line by choosing x and h randomly. Consider the case when h is fixed. It is not hard to check that for there is a partition of $\mathbb{F}_q^m = X_1 \cup \cdots \cup X_q$ where each X_i has size q^{m-1} such that $\cup_{x \in X_i} L_{x,h} = \mathbb{F}_q^m$. In other words:

Proposition 2 *The point line test has a partitioned query pattern.*

The proposed tolerant tester for $\mathcal{RM}(k, m, q)$ is as follows: pick $x, h \in \mathbb{F}_q^m$ uniformly at random and check if the input restricted to $L_{x,h}$ is τ-close to some univariate polynomial of degree k. If so accept, otherwise reject. When the threshold τ satisfies $\tau \leq 1 - \sqrt{k/q}$, the test can be implemented in polynomial time [11]. From Corollary 2, Proposition 2, Lemmas 2 and 3, the above is indeed a tolerant tester for $\mathcal{RM}(k, m, q)$ as recorded below.

Theorem 5 *For $0 \leq \tau \leq 1 - \sqrt{k/q}$ and $q = \Omega(k)$, $\mathcal{RM}(k, m, q)$ is (c_1, c_2, p) testable with $c_1 = \frac{n\tau}{3d}$, $c_2 = \frac{3(\tau+2)n}{d}$ and $p = n^{1-1/m}$ where $n = q^m$ and d are the block length and the distance of the code.*

6 Concluding remarks

Obtaining non-trivial lower bounds on the the block length of codes that are locally testable with very few (even 3) queries is an extremely interesting question. This problem has remained open and resisted even moderate progress despite all the advancements in constructions of LTCs. The requirement of having a tolerant local tester is a stronger requirement. While we have seen that we can get tolerance with similar parameters to the best known LTCs, it remains an interesting question whether the added requirement of tolerance makes the task of proving lower bounds more tractable. This seems like a good first step in making progress towards understanding whether asymptotically good locally testable codes exist, a question which is arguably one of the grand challenges in this area. For interesting work in this direction which proves that such codes, if they exist, cannot also be *cyclic*, see [3].

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the intractibility of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[2] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.

[3] L. Babai, A. Shpilka, and D. Stefankovic. Locally testable cyclic codes. In *Proceedings of 44th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 116–125, 2003.

[4] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. Vadhan. Robust PCPs of proximity, shorter PCPs and application to coding. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–10, 2004.

[5] E. Ben-Sasson and M. Sudan. Robust locally testable codes and products of codes. In *Proceedings of the 8th International Workshop on Randomization and Computation (RANDOM)*, pages 286–297, 2004.

[6] E. Ben-Sasson and M. Sudan. Simple PCPs with poly-log rate and query complexity. In *Proceedings of 37th ACM Symposium on Theory of Computing (STOC)*, 2005. To appear.

[7] E. Fischer and L. Fortnow. Tolerant versus intolerant testing for boolean properties. In *ECCC Technical Report TR04-105*, 2004.

[8] K. Friedl and M. Sudan. Some improvements to total degree tests. In *Proceedings of the 3rd Israel Symp. on Theory and Computing Systems (ISTCS)*, pages 190–198, 1995.

[9] O. Goldreich. Short locally testable codes and proofs (Survey). *ECCC Technical Report TR05-014*, 2005.

[10] O. Goldreich and M. Sudan. Locally testable codes and PCPs of almost linear length. In *Proceedings of 43rd Symposium on Foundations of Computer Science (FOCS)*, pages 13–22, 2002.

[11] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and Algebraic-geometric codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999.

[12] T. Kaufman and D. Ron. Testing polynomials over general fields. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 413–422, 2004.

[13] M. Parnas, D. Ron, and R. Rubinfeld. Tolerant property testing and distance approximation. In *ECCC Technical Report TR04-010*, 2004.

[14] A. Polishchuk and D. A. Spielman. Nearly-linear size holographic proofs. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 194–203, 1994.

[15] M. Sipser and D. Spielman. Expander codes. *IEEE Transactions on Information Theory*, 42(6):1710–1722, 1996.

[16] M. Sudan. *Efficient Checking of Polynomials and Proofs and the Hardness of Approximation Problems.* ACM Distinguished Theses Series. Lecture Notes in Computer Science, no. 1001, Springer, 1996.