Bounds on 2-Query Codeword Testing

Eli Ben-Sasson* Oded Goldreich† Madhu Sudan‡

April 3, 2003

Abstract

We present upper bounds on the size of codes that are locally testable by querying only two input symbols. For linear codes, we show that any 2-locally testable code with minimal distance δn over a finite field $\mathbb{F}$ cannot have more than $|\mathbb{F}|^{3/\delta}$ codewords. This result holds even for testers with two-sided error. For general (non-linear) codes we obtain the exact same bounds on the code size as a function of the minimal distance, but our bounds apply only for binary alphabets and one-sided error testers (i.e. with perfect completeness). Our bounds are obtained by examining the graph induced by the set of possible pairs of queries made by a codeword tester on a given code. We also demonstrate the tightness of our upper bounds and the essential role of certain parameters.

Keywords: Error-correcting codes, linear codes, sublinear-time algorithms, adaptivity,

*Division of Engineering and Applied Sciences, Harvard University and Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA. E-mail: `eli@eecs.harvard.edu`. Supported by NSF grants CCR-0133096, CCR-9877049, CCR 0205390, and NTT Award MIT 2001-04.

†Department of Computer Science, Weizmann Institute of Science, Rehovot, Israel. E-mail: `oded@wisdom.weizmann.ac.il`. Supported by the MINERVA Foundation, Germany.

‡Laboratory for Computer Science, Massachusetts Institute of Technology, 200 Technology Square, Cambridge, MA 02139. E-mail: `madhu@mit.edu`. Supported in part by NSF Awards CCR 9912342, CCR 0205390, and NTT Award MIT 2001-04.

1 Introduction

Locally testable codes are error-correcting codes that admit very efficient codeword testers. Specifically, using a constant number of (random) queries, non-codewords are rejected with probability proportional to their distance from the code.

Locally testable codes arise naturally from the study of probabilistically checkable proofs, and were explicitly defined in [3] and systematically studied in [5]. The task of testing a code locally may also be viewed as a special case of the general task of property testing initiated by [7, 4], where the property being tested here is that of being a codeword. In this paper we explore codes that can be tested with constant queries.

We focus on codes $\mathcal{C} \subset \Sigma^n$ that have large distance (i.e., each pair of codewords differ in at least $\Omega(n)$ coordinates) and large size (i.e., at the very least, $|\mathcal{C}|$ should grow with n and $|\Sigma|$). Such codes are known to exist. Specifically, in [5] locally testable codes are shown such that $|\mathcal{C}| = |\Sigma|^k$ for $k = n^{1-o(1)}$. We highlight two of these results:

1. For $\Sigma = \{0, 1\}$, *three* queries are shown to suffice. Furthermore, these codes are linear.
2. For $|\Sigma| > 2$, *two* queries are shown to suffice.[1]

This raises the question of whether locally testable codes can exist also for $\Sigma = \{0, 1\}$ and when considering two-query testers. In this paper, we show that the answer is essentially negative; that is, *for codes of linear distance, such codes can contain only a constant number of codewords.* More general statements are provided by Theorems 3.1 and 4.1, which address linear codes over arbitrary fields and non-linear binary codes, respectively. We also address the tightness of our upper-bounds and the essential role of certain parameters.

Organization: In Section 2 we present the main definitions used in this paper. In Section 3 we study *linear* codes that admit two-query codeword testers. In Section 4 we study general *binary* codes that admit two-query codeword testers (of perfect completeness). In Section 5 we show that our upper-bounds cease to hold already at ternary alphabets (rather than with the much larger alphabets considered in [5] and mentioned in Item 1 above). Finally, in Section 6, we show that perfect completeness is essential for the results regarding non-linear binary codes (presented in Section 4).

2 Formal Setting

We consider words over an alphabet Σ. For $w \in \Sigma^n$ and $i \in [n]$, we denote by w_i the i-th symbol of w; that is, $w = w_1 \cdots w_n$.

2.1 Codes

We consider codes $\mathcal{C} \subseteq \Sigma^n$ over a finite size alphabet Σ. The blocklength of $\mathcal{C}$ is n, and the size of $\mathcal{C}$ is its cardinality $|\mathcal{C}|$. We use normalized Hamming distance as our distance measure; that is, for $u, v \in \Sigma^n$ the distance $\Delta(u, v)$ is defined as the number of locations on which u and v differ,

[1] We comment that these codes are linear in a certain sense. Specifically, Σ is a vector space over a field F, and the code is a linear subspace over F (rather than over Σ). That is, if $\Sigma = F^\ell$ then $\mathcal{C} \subset \Sigma^n$ is a linear subspace of $F^{n \cdot \ell}$ (but not of Σ^n, no matter what finite field we associate with Σ). In the coding literature such codes are called F-linear. Indeed, a 2-query tester for an F-linear code over $\Sigma = F^\ell$ checks that the two symbols read (viewed as two sequences over F) satisfy a linear constraint over F.

divided by n (i.e., $\Delta(u,v) = |\{i : u_i \neq v_i\}|/n$). The relative minimal distance of a code, denoted $\delta(\mathcal{C})$, is the minimal normalized Hamming distance between two distinct codewords. Formally

$$\delta(\mathcal{C}) = \min_{u \neq v \in \mathcal{C}} \{\Delta(u,v)\}$$

The distance of a word w from the code, denoted $\Delta(w,\mathcal{C})$ is $\min_{v \in \mathcal{C}}\{\Delta(w,v)\}$.

A code is called redundant if its projection on some coordinate is constant (i.e., there exists $i \in \{1,\ldots,n\}$ such that for any two codewords w,w' it holds that $w_i = w'_i$). A redundant code can be projected on all non-redundant coordinates, yielding a code with the same size and distance, but smaller blocklength. Thus, w.l.o.g., *we assume all codes to be non-redundant*.

Typically (in this paper) Σ is a finite field $\mathbb{F}$ and we view $\mathbb{F}^n$ as a vector space over $\mathbb{F}$. In particular, for $u,v \in \mathbb{F}^n$ the of the two is $\langle v,u \rangle = \sum_{i=1}^n v_i \cdot u_i$ (all arithmetic operations are in $\mathbb{F}$). The weight of $v \in \mathbb{F}^n$, denoted $\mathrm{wt}(v)$, is the number of non-zero elements in v. In this case $\Delta(u,v) = \mathrm{wt}(u-v)/n$.

2.2 Testers and tests

By a codeword tester (or simply tester) with query complexity q, completeness c and soundness s (for the code $\mathcal{C} \subseteq \Sigma^n$) we mean a *randomized* oracle machine that given oracle access to $w \in \Sigma^n$ (viewed as a function $w : \{1,\ldots,n\} \to \Sigma$) satisfies the following three conditions:

- Query Complexity q: The tester makes at most q queries to w.
- Completeness: For any $w \in \mathcal{C}$, given oracle access to w the tester accepts with probability at least c.
- Soundness: For any w that is at relative distance at least $\delta(\mathcal{C})/3$ from $\mathcal{C}$, given oracle access to w, the tester accepts with probability at most s.[2]

If $\mathcal{C}$ has a codeword tester with query complexity q, completeness c and soundness s we say $\mathcal{C}$ is $[q,c,s]$-locally testable.

A deterministic test (or simply test) with query complexity q is a *deterministic* oracle machine that given oracle access to $w \in \Sigma^n$ makes at most q queries to w, and outputs 1 (= accept) or 0 (= reject). Any (randomized) tester can be described as a distribution over deterministic tests, and we adopt this view throughout the text.

A (deterministic) test is called adaptive if its queries depend on previous answers of the oracle, and otherwise it is called non-adaptive. A test has perfect completeness if it accepts all codewords. Both notions extend to (randomized) testers. Alternatively, we say that a tester is non-adaptive (resp., has perfect completeness) if all the deterministic tests that it uses are non-adaptive (resp., have perfect completeness resp.), and otherwise it is adaptive (resp., has non-perfect completeness).

3 Linear Codes

In this section we show that $[2,c,s]$-locally testable *linear* codes with constant minimal relative distance must have very small size. Throughout this section $\mathbb{F}$ is a finite field of size $|\mathbb{F}|$. A code $\mathcal{C} \subseteq \mathbb{F}^n$ is called linear if it is a linear subspace of $\mathbb{F}^n$. The main result of this section is the following.

[2]We have set the *detection radius* of the tester at third its distance (i.e., for any w whose distance from $\mathcal{C}$ is at least $\frac{1}{3} \cdot \delta(\mathcal{C})$ the test rejects with probability at least s). As will be evident from the proofs, our results hold for any radius less than half the distance.

Theorem 3.1 *Let $\mathcal{C} \subset \mathbb{F}^n$ be a $[2, c, s]$-locally testable* linear *code with minimal relative distance δ. If $c > s$ then*

$$|\mathcal{C}| \leq |\mathbb{F}|^{3/\delta}$$

We start by pointing out that, when considering testers for linear codes, the tester can be assumed to be non-adaptive and with perfect completeness. This holds by the following result of [2].

Theorem 3.2 [2]: *If a linear code (over any finite field) is $[q, c, s]$-locally testable using an adaptive tester, then it is $[q, 1, 1 - (c - s)]$-locally testable using a non-adaptive tester.*

Notice that if we start off with a tester having completeness greater than soundness ($c > s$), then the resulting non-adaptive, perfect-completeness tester (guaranteed by Theorem 3.2) will have soundness strictly less than 1. Thus, in order to prove Theorem 3.1 it suffices to show the following.

Theorem 3.3 *Let $\mathcal{C} \subseteq \mathbb{F}^n$ be a $[2, 1, s]$-locally testable linear code, with $s < 1$, and let the minimal relative distance be δ. Then:*

$$|\mathcal{C}| \leq |\mathbb{F}|^{3/\delta}$$

In the rest of the section we prove Theorem 3.3. The proof idea is as follows. Each possible test of query complexity 2 and perfect completeness imposes a constraint on the code, because all codewords must pass the test. Thus, we view the n codeword coordinates as *variables* and the set of tests as inducing constraints on these variables (i.e., codewords correspond to assignments (to the variables) that satisfy all these constraints). Since the code is linear, each test imposes a *linear* constraint on the pair of variables queried by it. (A linear constraint on the variables x, y has the form $ax + by = 0$ for some fixed $a, b \in \mathbb{F}$). We will show that in a code of large distance, these constraints induce very few satisfying assignments. Specifically, we look at the graph in which the vertices are the (n) codeword-coordinates (or variables) and edges connect two vertices that share a test. The main observation is that in any codeword, the value of all variables in a connected component are determined by the value of any one variable in the component; that is, the assignment to a single variable determines the assignment to the whole component. By perfect completeness, any word that satisfies all constraints in all connected components will pass *all* tests. Hence there cannot be many variables in small connected components, for then we could find a word that is far from the code and yet is accepted with probability 1. But this means that the code is essentially determined by the (small number of) large connected components, and hence the size of the code is small. We now give the details, starting with a brief discussion of *dual codes* which is followed by the proof.

3.1 Linearity Tests and Dual Codes

Recall that $\mathcal{C} \subseteq \mathbb{F}^n$ is linear iff for all $u, v \in \mathcal{C}$ we have $u + v \in \mathcal{C}$. In this case $\delta(\mathcal{C}) = \min_{w \in \mathcal{C}}\{\mathrm{wt}(w)/n\}$. As pointed out in [6], codeword tests for linear codes are intimately related to the "dual" of the code. For a linear code $\mathcal{C}$, the *dual code* $\mathcal{C}^\perp$ is defined as the subspace of $\mathbb{F}^n$ orthogonal to $\mathcal{C}$, i.e.

$$\mathcal{C}^\perp = \{v : v \perp \mathcal{C}\}$$

where $v \perp \mathcal{C}$ iff for all $u \in \mathcal{C}$, $v \perp u$ (recall $v \perp u$ iff $\langle v, u \rangle = 0$).

The support of a vector v, denoted $\mathbf{Supp}(v)$, is the set of indices of non-zero entries. Similarly, the support of a test T is the set of indices it queries. Notice that a non-adaptive test with query complexity q has support size q. For $v, u \in \mathbb{F}^n$ we say that v covers u if $\mathbf{Supp}(v) \supseteq \mathbf{Supp}(u)$. A test is called trivial if it always accepts. Elementary linear algebra gives the following claim.

Proposition 3.4 *The support of any* non-trivial perfect-completeness test *for $\mathcal{C}$ covers an element of $\mathcal{C}^\perp \setminus \{0^n\}$.*

Clearly one can assume that all tests used by a tester are non-trivial. We also assume $\mathcal{C}^\perp$ has no element of weight 1, because otherwise $\mathcal{C}$ is redundant. Since we consider only testers that make two queries, it follows that all tests they use have support size exactly two.

3.2 Upper Bounds on Code Size

By the above discussion (i.e., end of Section 3.1), we may assume (w.l.o.g.) that the $[2,1,s]$-tester for $\mathcal{C}$ is described by a distribution over

$$\mathcal{C}_2^\perp \stackrel{\text{def}}{=} \{v \in \mathcal{C}^\perp : \text{wt}(v) = 2\}$$

The test corresponding to $v \in \mathcal{C}_2^\perp$ refers to the orthogonality of v and the oracle w; that is, the test accepts w if $v \perp w$ and rejects otherwise.[3] We now look at $\mathcal{C}_2^\perp$ and bound the size of $(\mathcal{C}_2^\perp)^\perp$. Our theorem will follow because $\mathcal{C} \subseteq (\mathcal{C}_2^\perp)^\perp$.

The set $\mathcal{C}_2^\perp$ gives rise to a natural graph, denoted $G_\mathcal{C}$. The vertex set of $G_\mathcal{C}$ is $V(G_\mathcal{C}) = \{1, \ldots, n\}$ and $(i,j) \in E(G_\mathcal{C})$ iff there exists $v_{ij} \in \mathcal{C}_2^\perp$ with $\mathbf{Supp}(v_{ij}) = \{i, j\}$.

The key observation is that, for any edge $(i,j) \in E(G_\mathcal{C})$ there is some $c_{ij} \in \mathbb{F} \setminus \{0\}$ such that for any $w \in \mathcal{C}$ it holds that $w_i = c_{ij} \cdot w_j$. To see this, notice the constraint corresponding to (i,j) can be written as $a_{ij}w_i + b_{ij}w_j = 0$, where $a_{ij}, b_{ij} \in \mathbb{F} \setminus \{0\}$ (if either a_{ij} or b_{ij} are 0 then v_{ij} has support size one, meaning $\mathcal{C}$ is redundant). So, by transitivity, the value of w on all variables in the connected component of i, is determined by w_i. (Moreover, all these values are non-zero iff $w_i \neq 0$.) Assuming that the number of connected components is k, this implies that there can be at most $|\mathbb{F}|^k$ different codewords (because there are only k degrees of freedom corresponding to the settings (of all variables) in each of the k components). To derive the desired bound we partition the components into big and small ones, and bound the number of codewords as a function of the number of big components (while showing that the small components do not matter).

Let $C_1, \ldots, C_k$ be the connected components of $G_\mathcal{C}$. We call a component small if its cardinality is less than $\delta n/3$. Without loss of generality, let $C_1, \ldots C_s$ be all the small components, and let $S = \bigcup_{i=1}^s C_i$ denote their union.

Claim 3.5 $|S| \leq 2\delta n/3$.

Proof: Otherwise there exists $I \subset \{1, \ldots, s\}$ such that

$$\delta n/3 \leq \sum_{i \in I} |C_i| < 2\delta n/3$$

For every $i \in I$, we consider a vector $w^i \in (\mathcal{C}_2^\perp)^\perp$ with $\mathbf{Supp}(w^i) = C_i$. To see that such a vector exist, set an arbitrary coordinate of C_i to 1 (which is possible because the code is not redundent) and force non-zero values to all other coordinates in C_i (by virtue of the above discussion). Furthermore, note that this leaves all coordinates out of C_i unset, and that the resulting w^i satisfy all tests in

[3]Notice that since $\text{wt}(v) = 2$ such a test amounts to two queries into w.

$\mathcal{C}_2^{\perp}$ (where the test that correspond to the edges in C_i are satified by our setting of the non-zero values, whereas all other tests refer to vertices out of C_i and are satisfied by zero values). Now, define $w = \sum_{i \in I} w^i$. By definition, we have $\mathbf{Supp}(w) = S$, and $\delta n/3 \leq \mathrm{wt}(w) < 2\delta n/3$ follows by the hypothesis. Hence, $\Delta(w, \mathcal{C}) \geq \delta/3$.

On the other hand, w is orthogonal to $\mathcal{C}_2^{\perp}$. To see this, consider any $v \in \mathcal{C}^{\perp}$. If $\mathbf{Supp}(v) \subseteq C_i$, for some $i \in I$, then the "view v has of w" (i.e. the values of the coordinates v queries) is identical to the view v has of the codeword w^i, and so $\langle v, w \rangle = \langle v, w^i \rangle = 0$. Otherwise (i.e., $\mathbf{Supp}(v)$ has empty intersection with S), by definition v "sees" only zeros, and so $\langle v, w \rangle = 0$.

We conclude w is $\frac{\delta}{3}$-far from $\mathcal{C}$, yet it passes all possible tests of query complexity two. This contradicts the soundness condition, and the claim follows. $\square$

Proof (of Theorem 3.3): *Assume for the sake of contradiction* that

$$|\mathcal{C}| > |\mathbb{F}|^{3/\delta}$$

Recall that (by the "key observation") the values of all variables in a connected component are determined by the value of a single variable in this component. Since there are at most $3/\delta$ large connected components in $G_{\mathcal{C}}$ (because each has cardinality at least $\delta n/3$), the contradiction hypothesis implies that there exist two codewords $x \neq y$ that agree on all variables that reside in the large connected components. Indeed, these two codewords $x \neq y$, may differ on variables that reside in the small connected components (i.e., variables in S), but Claim 3.5 says that there are few such variables (i.e.. $|S| \leq 2\delta n/3$). By linearity $x - y \in \mathcal{C}$ (but $x - y \neq 0^n$), and so $0 < \mathrm{wt}(x - y) \leq |S| < \delta n$. We have reached a contradiction (because $\mathcal{C}$ has distance δ), and Theorem 3.3 follows. ▮

3.3 Tightness of the Upper Bound

We remark that our upper bound is quite tight. For any $\delta < 1$, consider the following code $\mathcal{C}_n \subset \mathbb{F}^n$ formed by taking $1/\delta$ elements of $\mathbb{F}$ and repeating each one of them δn times. Thus, a codeword in $\mathcal{C}_n$ is formed of $1/\delta$ blocks, each block of the form $e^{\delta n}$ for some $e \in \mathbb{F}$ (here e^k means k repetitions of e).

Proposition 3.6 *$\mathcal{C}_n$ is a linear $[2, 1, 1 - \frac{2\delta}{3|\mathbb{F}|}]$-locally testable code with minimal relative distance δ and size $|\mathbb{F}|^{1/\delta}$.*

For instance, taking $\mathbb{F} = GF(2)$, the soundness parameter in the proposition is $1 - \delta/3$.

Proof: The linearity, distance and size of $\mathcal{C}_n$ are self-evident. Consider the following natural tester for $\mathcal{C}_n$: Select a random block, read two random elements in it, and accept iff the two are equal. This tester has perfect completeness and query complexity 2. As to the soundness, let $k = 1/\delta$ and write $v \in \mathbb{F}^n$ as $(v^{(1)}, \ldots, v^{(k)})$, where $v^{(i)}$ is the i-th block of v (i.e., $|v^{(i)}| = \delta n$). The Hamming distance of v from $\mathcal{C}_n$ is the sum of the Hamming distances of the individual blocks $v^{(i)}$ from the code $B = \{e^{\delta n} : e \in \mathbb{F}\}$.

Suppose v has relative distance at least $\delta/3$ from $\mathcal{C}_n$. Let δ_i denote the relative distance of $v^{(i)}$ from B. Then, $\frac{1}{k}\sum_{i=1}^{k} \delta_i \geq \delta/3$ (and $\delta_i \leq 1 - \frac{1}{|\mathbb{F}|}$). The acceptance probability of the tester equals

$$\frac{1}{k}\sum_{i=1}^{k}\left(\delta_i^2 + (1-\delta_i)^2\right) \quad = \quad 1 - \frac{2}{k}\sum_{i=1}^{k}(1-\delta_i)\cdot\delta_i$$

$$\leq 1 - \frac{2}{k|\mathbb{F}|}\sum_{i=1}^{k}\delta_i$$
$$\leq 1 - \frac{2\delta}{3|\mathbb{F}|}$$

where the first inequality is due to $\delta_i \leq 1 - \frac{1}{|\mathbb{F}|}$. Thus, the soundness parameter is as claimed. □

4 Non-Linear Codes

In this section we provide upper bounds on the code size of arbitrary (i.e., possibly non-linear) 2-locally testable codes. Our bounds apply only to binary codes and testers with perfect completeness, and with good reason: There exist good 2-testable binary codes with non-perfect completeness (see Section 6) and there exist good 2-testable codes with perfect completeness over ternary alphabets (see Section 5). Our main result is:

Theorem 4.1 *If $\mathcal{C} \subseteq \{0,1\}^n$ is a $[2,1,s]$-locally testable code with minimal relative distance δ and $s < 1$, then*

$$|\mathcal{C}| \leq 2^{3/\delta}$$

The proof (presented below) generalizes that of the *binary* linear case (binary means $\mathbb{F} = GF(2)$), with some necessary modifications, which we briefly outline now. In the binary linear case a test querying x_i and x_j forces $x_i = x_j$ for all codewords (this is the only possible linear constraint of size two over $GF2$). In that case, the set of all tests corresponds to an undirected graph in which each connected component forces all variables to have the same value. In the non-linear case a test (adaptive or non-adaptive) corresponds to a 2-CNF. (Recall that in both cases we deal with perfect completeness testers.) The set of all tests (which is itself a 2-CNF) corresponds to a *directed* graph of constraints on codewords, where the constraint $x_i \vee x_j$ translates to the pair of directed edges $\bar{x}_i \to x_j$ and $\bar{x}_j \to x_i$. In the resulting *directed* graph, a *strongly* connected component takes the role played by the connected component in the linear case. Namely, for any codeword, all variables in a strongly connected component are fixed by the value of a single variable in the component. As in the linear case, we use the properties of the code and its tester (i.e., the code's large distance and the fact that the tester rejects any word that is far from the code with non-zero probability) to show that the weight of the *small* strongly connected components is small. Hence, the code is determined by a small number of *large* connected components.

Proof of Theorem 4.1

Again, we view the n codeword coordinates as *variables* and the set of tests (which are 2-CNFs) as inducing constraints on these variables. Let $\mathcal{F}$ be the conjunction of all non-trivial deterministic tests that are used by a 2-query tester that has perfect completeness with respect to $\mathcal{C}$. We look at the satisfying assignments of $\mathcal{F}$, and use this to bound the size of $\mathcal{C}$. If $\mathcal{F}$ includes a clause of size 1 then $\mathcal{C}$ is redundant. Thus, assuming non-redundancy of $\mathcal{C}$ implies that $\mathcal{F}$ is can be represented by a 2-CNF in which each clause has *exactly* two literals.

We examine the following directed graph $G_{\mathcal{F}}$. The vertex set of $G_{\mathcal{F}}$ is the set of literals $\{x_1, \bar{x}_1 \ldots, x_n, \bar{x}_n\}$. For each clause $(\ell \vee \ell') \in \mathcal{F}$ we introduce in $G_{\mathcal{F}}$ one directed edge from $\bar{\ell}$ to ℓ', and one from $\bar{\ell}'$ to ℓ. We use the notation $\ell \rightsquigarrow \ell'$ to indicate the existence of a directed path from

ℓ to ℓ' in $G_{\mathcal{F}}$. We use the notation $w(\ell)$ to denote the value of literal ℓ under assignment w to the underlying variables. Identifying $True$ with 1 and $False$ with 0, we have

Claim 4.2 (folklore): *The following two conditions are equivalent*

1. *The assignment w satisfies $\mathcal{F}$.*
2. *For every directed edge $\ell \leadsto \ell'$ it holds that $w(\ell) \leq w(\ell')$.*

A strongly connected component in a directed graph G is a maximal set of vertices $C \subseteq V(G)$ such that for any $v, v' \in C$ it holds that $v \leadsto v'$. For two strongly connected components C and C' in G, we say $C \leadsto C'$ iff there exist $v \in C$ and $v' \in C'$ such that $v \leadsto v'$. (Indeed, this happens iff for all $v \in C, v' \in C'$ it holds that $v \leadsto v'$.)

By Claim 4.2, w satisfies all constraints corresponding to edges of a strongly connected component C iff $w(\ell) = w(\ell')$ for all $\ell, \ell' \in C$. So, any satisfying assignment w either sets to 1 *all* literals in C, or sets them all to 0. In the first case we say that $w(C) = 1$ and in the latter we say $w(C) = 0$.

Let L be the set of literals belonging to large strongly-connected components, where a component is called large iff its cardinality is at least $\delta n/3$. Consider an arbitrary assignment ρ' to the variables of L that can be extended to a satisfying assignment (to $\mathcal{F}$). In particular, ρ' does not falsify any clause of $\mathcal{F}$ (i.e., no clause of $\mathcal{F}$ is set to 0 by ρ'). A literal $\ell \notin L$ is said to be forced by ρ' if there exists $\ell' \in L$ such that $\ell' \leadsto \ell$ and $\rho'(\ell') = 1$. This is because any satisfying assignment to $\mathcal{F}$ that extends ρ' must set ℓ to 1 (since for such an assignment ρ it must holds that $\rho(\ell) \geq \rho(\ell') = 1$. Indeed, the complementary literal (i.e., $\bar{\ell}$) is forced to 0. Let ρ be the *closure* of ρ' obtained by (iteratively) fixing all forced literals to the value 1 (and their complementary literals to 0). By definition, ρ does not falsify $\mathcal{F}$. Let S_ρ be the set of unfixed variables under ρ.

Claim 4.3 *For any closure ρ of an assignment that satisfies L, it holds that $|S_\rho| \leq 2\delta n/3$.*

Proof: Otherwise, let $C_1, \ldots, C_k$ be a topological ordering of the unfixed strongly connected components comprising S_ρ, where the ordering is according to $\leadsto$ (as defined above). (Indeed, the digraph defined on the C_i's by $\leadsto$ is acyclic.) For $j = 0, \ldots, k$, let $v^{(j)}$ be the assignment extending ρ defined by:

$$v^{(j)}(C_i) = \begin{cases} 0 & i \leq j \\ 1 & i > j \end{cases}$$

By Claim 4.2, each assignment $v^{(j)}$ satisfies $\mathcal{F}$. Since $\mathcal{C}$ is 2-locally testable with soundness $s < 1$, each word that is at distance at least $\delta/3$ from $\mathcal{C}$ must falsify some clause in $\mathcal{F}$. But since $v^{(j)}$ satisfies $\mathcal{F}$, it must be that $v^{(j)}$ is within distance $\delta/3$ from some codeword, denoted $w^{(j)}$. By the constradiction hypothesis, we have $\Delta(v^{(0)}, v^{(k)}) = |S_\rho|/n > 2\delta/3$, which implies $w^{(0)} \neq w^{(k)}$ (because $\Delta(v^{(0)}, v^{(k)}) \leq \Delta(v^{(0)}, w^{(0)}) + \Delta(w^{(0)}, w^{(k)}) + \Delta(w^{(k)}, v^{(k)})$, which is upper-bounded by $2 \cdot (\delta/3) + \Delta(w^{(0)}, w^{(k)})$). It follows that

$$\Delta(v^{(k)}, w^{(0)}) \geq \Delta(w^{(k)}, w^{(0)}) - \Delta(w^{(k)}, v^{(k)}) \geq \delta - (\delta/3) = 2\delta/3$$

On the other hand, recall that $\Delta(v^{(0)}, w^{(0)}) \leq \delta/3$. Since, for each j, it holds that $\Delta(v^{(j)}, v^{(j+1)}) < \delta/3$ (because $|C_j| < \delta n/3$), there must be $j \in \{0, 1, \ldots, k\}$ such that $\delta/3 \leq \Delta(v^{(j)}, w^{(0)}) \leq 2\delta/3$. For this j, it holds that $\Delta(v^{(j)}, \mathcal{C}) \geq \delta/3$. But $v^{(j)}$ satisfies $\mathcal{F}$ and so it is accepted by the tester with probability 1, in contradiction to the soundness condition. $\square$

Our proof is nearly complete. As in the proof of Theorem 3.3, assume for the sake of contradiction that

$$|\mathcal{C}| > 2^{\delta/3}$$

In this case, there must be two distinct codewords $w \neq u$ that agree on all large connected components. Let ρ' be the restriction of w to the variables of the large connected components. That is, ρ' agrees with w and with u on the assignment to all variables in L and is unfixed otherwise. Let ρ be the closure of ρ' (obtained by forcing as above). Note that w and u are satisfying assignments to $\mathcal{F}$ that agree on ρ', so they also must agree on ρ (which is forced by ρ'). Thus, by Claim 4.3

$$0 < \Delta(u, w) \leq |S_\rho|/n < \delta$$

This contradicts the hypothesis that the minimal distance of $\mathcal{C}$ is δ, and the theorem follows. ∎

5 Ternary Alphabets

In this section we show that our general upper-bound (i.e., Theorem 4.1) ceases to hold already at ternary alphabets. That is, we present (non-linear) *ternary* codes admitting two-query testers. We also show that, in contrast to Theorem 3.1 (which refers to linear codes), there exist good $GF(2)$-linear codes over the alphabet $\{0,1\}^2$ that admit two-query testers.[4] The latter construction is postponed to Section 5.4, and we first focus on the construction of ternary codes.

5.1 Outline

In this section we present $[2, 1, s]$-locally testable codes, over ternary alphabets, that have linear distance[5] and non-trivial rate. We start with a good linear $[3, 1, s]$-locally testable code $\mathcal{C} \subset \{0,1\}^n$ over the binary alphabet (cf. [5]), where each test is defined by a parity constraint over three variables. We replace each parity by the four clauses that encode it as a 3-CNF, where the variables correspond to the bits of the alleged codeword. Let F be the set of all clauses used by our tester for $\mathcal{C}$. We think of $\mathcal{C}$ as the set of satisfying assignments to F. Our new ternary code $\mathcal{C}'$ is formed by appending to each codeword $w \in \mathcal{C}$ a list $L \in \{0,1,2\}^F$, which for each clause $C \in F$ specifies the literal satisfying C under the assignment w. Our new tester T' for $\mathcal{C}'$ selects a random clause from F, reads the index of the purported satisfying literal, and verifies that w indeed assigns to this literal the value that satisfies C. This naive strategy faces a couple of problems that we briefly discuss and show how to solve.

- The most obvious problem is that (as defined above) $\mathcal{C}'$ has distance 1. The reason is that, for any $w \in \mathcal{C}$, there are many lists $L \in \{0,1,2\}^F$ such that $(w, L) \in \mathcal{C}'$, and some of these lists differ only in one location. Specifically, if w satisfies more than one of the literals of a clause $C \in F$ (which is the case for any $w \in \mathcal{C}$ and some of $C \in F$), then we can choose to put in our list (at the position corresponding to C) the index of any of these literals.

  This problem is resolved by enforcing a *unique* list L_w for each $w \in \mathcal{C}$; specifically, we require that, for each clause, the list includes the smallest index of a satisfiable literal in the clause. Furthermore, we enforce the use of this unique list by augmenting the tester such that it

[4] Recall that $GF(2)$-linear codes over the alphabet $\{0,1\}^2$ are different from linear codes over $\{0,1\}^2$. See discussions in Footnote 1 and in Section 5.4.

[5] To get meaningful results on local testability one must assume large minimal distance. Otherwise, the trivial code $\mathcal{C} = \{0,1\}^n$ has rate 1 and is $[0, 1, 0]$-locally testable.

sometimes emulates the basic tester (outlined above) and otherwise verifies the uniqueness property. Specifically, the new tester sometimes checks that the purported literal satisfies the clause, and sometimes checks that a literal with a smaller index does not satisfy this clause.

- A second obvious problem is that the blocklength of the new code is likely to be dominated by the size of the appended list, whereas the distance might be as small as that of the original code (here we refer to the non-relative distance). Indeed, for $\mathcal{C} \subset \{0,1\}^n$, the tester for $\mathcal{C}$ might use as many as $\binom{n}{3}$ distinct tests (and thus we may have $|F| = \Theta(n^3)$). While $\mathcal{C}$ has good minimal distance, we have no similar guarantee on the distance between unique lists of distinct codewords in $\mathcal{C}$. For all we know there might be two *distinct* words in $\mathcal{C}$ that share the same unique list. Thus, we might end up with a code of blocklength $\approx n^3$ and minimal (absolute) distance δn. There is not much hope for showing the local testability of such a code, because changing $\delta n/3$ symbols in the unique list of a codeword w will hardly be noticed by our tester.

  This problem is easily resolved by repeating each bit of the original code sufficiently many times (i.e., $\Theta(|F|/n)$ repetitions will do). Indeed, we have to check that the corresponding part of the new codeword is of the right format; that is, we sometimes check that two random copies of the same bit-location hold the same value.

  To improve the rate[6] of the new code (i.e., maintain the rate of the original code), we will first modify the given tester so that it only uses $O(n)$ (rather than $O(n^3)$) distinct tests.

We comment that an alternative approach towards resolving the second problem may capitalize on the fact that the unique valid lists associated with codewords of the original code must be far apart from one another. This fact can be proved by observing that a random test (of the original tester) is likely to read different values from the two codewords, and thus some of the corresponding four clauses are (uniquely) satisfied by literals of different index.

5.2 Uniform Testers

Our starting point is the following construction of [5], which gives linear locally testable codes over the binary alphabet with linear distance and pretty good rate. In what follows, it will be convenient to use a tester with detection radius $\delta/4$ (where δ is the minimal distance of the code), so we state the theorem with this additional property.

Theorem 5.1 [5, Prop. 5.9]: *There exist constants $1 > s, \delta > 0$ such that for arbitrarily large integers n there exist linear $[3,1,s]$-locally testable codes $\mathcal{C}_n \subseteq \{0,1\}^n$, with relative minimal distance at least δ and size $|\mathcal{C}_n| = 2^{n/\exp(\log^{0.51} n)}$. Moreover, there exists a tester T for this code, such that any word that is (at least) $\delta/4$-far from $\mathcal{C}$ is accepted by T with probability at most s (and words in $\mathcal{C}$ are accepted with probability 1).*

Recall that a tester is a distribution over tests. Our first observation is that this distribution can be assumed to be uniform over a set of $O(n)$ tests, in which case we say that tester is uniform. This fact was established in [5, Prop. 5.8]. (Actually, as in our case, [5, Prop. 5.8] was established as a preparatory step towards establishing [5, Prop. 5.9].) From now on we assume the tester for the $[3,1,s]$-locally testable code $\mathcal{C}_n \subseteq \{0,1\}^n$ is uniform and the number of tests m is linear in n.

[6] The rate of $\mathcal{C} \subset \{0,1\}^n$ is defined as $(\log_2 |\mathcal{C}|)/n$.

5.3 Ternary Code Construction and Analysis

Recall that $\mathcal{C} \stackrel{\text{def}}{=} \mathcal{C}_n \subset \{0,1\}^n$ is a (binary) linear $[3,1,s]$-locally testable code over the binary alphabet that has a uniform tester over a set of $m = cn$ tests, where each test is a parity check of three variables (corresponding to locations in the codeword). Following the discussion in Section 3.1, we replace each such parity test by the four corresponding clauses (each of size 3), and let F denote the resulting sequence of m clauses. We view an element $v \in \{0,1\}^n$ as an assignment to the CNF F. The perfect completeness of the tester means every $w \in \mathcal{C}_n$ is a satisfying assignment of F. Now, for some constant integer $c' > c$ (to be determined), we consider the following code.

Construction 5.2 (Our Ternary Code): *Let $\mathcal{C}' \subset (\{0,1\}^n)^{c'} \times \{1,2,3\}^{4m}$, viewed as a subset of $\{0,1,2\}^{c'n+cn}$, be the set of all pairs of the form $((v^{(1)},\ldots,v^{(c')}),(\ell_1,\ldots,\ell_{4m}))$ for which there exists $w \in \mathcal{C}$ such that the following two conditions holds*

1. *$(v^{(1)},\ldots,v^{(c')})$ is a repetitive encoding of w; that is, for $i = 1,\ldots,c'$, it holds that $v^{(i)} = w$.*
2. *$(\ell_1,\ldots,\ell_{4m})$ indicates the lexicographically first sequence of literals that satisfies F; that is, for every $j = 1,..,4m$, it holds that the ℓ_j-th literal of C_j satisfies this clause whereas for every $i < \ell_j$ the i-th literal of this clause does not satisfy it.*

Recall that $m = cn$.

Note that the absolute distance of $\mathcal{C}'$ is lower-bounded by c' times the absolute distance of $\mathcal{C}$. Thus, the relative distance $\delta(\mathcal{C}')$ is lower bounded by $c' \cdot \delta(\mathcal{C}) \cdot n/(c'n + cn)$, which in turn is greater than $\delta(\mathcal{C})/2$ (because $c' > c$). In fact, we can achieve $\delta(\mathcal{C}') \geq (1-\gamma)\cdot\delta(\mathcal{C})$ for any $\gamma > 0$, by selecting c' to be a sufficiently large constant. Under this setting, $\mathcal{C}'$ maintains the asymptotic rate of $\mathcal{C}$ up to a constant factor, because

$$\frac{\log_3|\mathcal{C}'|}{c'n+cn} = \frac{1}{(c'+c)\log_2 3}\cdot\frac{\log_2|\mathcal{C}|}{n} > \frac{1}{2(c'+c)}\cdot\frac{1}{\exp(\log^{0.51} n)}$$

and the loss of a factor of $2(c'+c)$ in the rate is not significant.

Construction 5.3 (Our tester $T'((v^{(1)},\ldots,v^{(c')}),(\ell_1,\ldots,\ell_{4m}))$): *We assume, for simplicity, that $(v^{(1)},\ldots,v^{(c')}) \in \{0,1\}^{c'n}$ and $(\ell_1,\ldots,\ell_{4m}) \in \{1,2,3\}^{4m}$. This assumption can be enforced by an adequate translation.*

1. *With probability $1/2$, perform a random test of proper repetitive encoding of a binary string. That is, we check if a random variable is assigned the same boolean value by two random copies. Specifically, select uniformly copies $i_1, i_2 \in [c']$ and a variable $j \in [n]$, and accept if and only if the j-th symbol of $v^{(i_1)}$ equals the j-th symbol of $v^{(i_2)}$ and this symbol is in $\{0,1\}$.*
2. *With probability $1/2$, perform a random test of lexicographically-first satisfaction. That is, we check if a random variable in a random clause is assigned* (in a random copy) *a value that is consistent with the lexicographically-first satisfying variable as indicated by the list. Specifically, uniformly select a clause $j_1 \in [m]$, a position in it $j_2 \in [3]$, and a copy $i \in [c']$, let $\mathrm{idx}(j_1,j_2) \in [n]$ be the index of the variable that appears as the j_2-th variable in C_{j_1}, obtain the value $v \leftarrow v^{(i)}_{\mathrm{idx}(j_1,j_2)} \in \{0,1\}$ and the indicator $\ell \leftarrow \ell_{j_1} \in [3]$, and accept if and only if the following holds*

(a) *In case $j_2 = \ell$, we accept if and only if the i_2-th literal in C_{j_1} is satisfied when assigning v to the corresponding variable.*

(b) *In case $j_2 < \ell$, we accept if and only if the i_2-th literal in C_{j_1} is not satisfied when assigning v to the corresponding variable.*

(c) *In case $j_2 > \ell$, we always accept.*

Notice that T' has query complexity 2 (and is non-adaptive). It is also easy to see that it has perfect completeness. It is left to analyze the soundness of T'.

Lemma 5.4 *For some constant $s' < 1$ the following holds. For every $((v^{(1)}, \ldots, v^{(c')}), (\ell_1, \ldots, \ell_{4m}))$ that is at relative distance at least $\delta(\mathcal{C}')/3$ from $\mathcal{C}'$, the tester T' accepts $((v^{(1)}, \ldots, v^{(c')}), (\ell_1, \ldots, \ell_{4m}))$ with probability at most s'.*

Proof: In the following proof we refer to distances of various strings from three different codes, having different blocklength. Thus, it will be less confusing to refer to absolute distnaces (rather than to relative ones). Thus, throughout this proof, we always make reference also to absolute distance.

We consider two cases with respect to the distance of $(v^{(1)}, \ldots, v^{(c')})$ from the repetition code. For some $\epsilon > 0$ (to be determined below), the easy case is when $(v^{(1)}, \ldots, v^{(c')})$ is ϵ-far from being a repetitive encoding of any binary string (i.e., $(v^{(1)}, \ldots, v^{(c')}) \in \{0,1\}^{c'n}$ is at distance at least $\epsilon c'n$ from the repetition code). In this case, by virtue of the first sub-tester of T', the sequence $((v^{(1)}, \ldots, v^{(c')}), (\ell_1, \ldots, \ell_{4m}))$ is rejected with constant probability.

Otherwise, $(v^{(1)}, \ldots, v^{(c')})$ is ϵ-close to such a repetitive encoding of some string $w \in \{0,1\}^n$ (i.e., $(v^{(1)}, \ldots, v^{(c')}) \in \{0,1\}^{c'n}$ is at distance at most $\epsilon c'n$ from $w^{c'}$ (i.e., w repeated C' times)). This means that at least half of the $v^{(i)}$'s are at distance at most $2\epsilon n$ from w. On the other hand, since $(\ell_1, \ldots, \ell_{4m})$ accounts for $4cn$ locations in the code $\mathcal{C}'$ and $((v^{(1)}, \ldots, v^{(c')}), (\ell_1, \ldots, \ell_{4m}))$ is at distance at least $d \stackrel{\text{def}}{=} \frac{\delta(\mathcal{C}')}{3} \cdot (c'n + 4cn)$ from $\mathcal{C}'$, it follows that $(v^{(1)}, \ldots, v^{(c')})$ is at distance at least $d - 4cn$ from some repetitive encoding of a codeword of $\mathcal{C}$. Thus, by the triangle inequality, the distance of w from $\mathcal{C}$ must be at least

$$\begin{aligned} & \frac{\min_{u \in \mathcal{C}}\{\mathrm{dist}((v^{(1)}, \ldots, v^{(c')}), u^{c'})\} - \mathrm{dist}((v^{(1)}, \ldots, v^{(c')}), w^{c'})}{c'} \\ \geq\ & \frac{d - 4cn - \epsilon c'n}{c'} \\ >\ & \left(\frac{\delta(\mathcal{C}')}{3} - \frac{4c}{c'} - \epsilon\right) \cdot n \qquad (1) \end{aligned}$$

We take c' large enough and ϵ small enough, so that Eq. (1) is greater than $(\frac{\delta(\mathcal{C})}{4} + 2\epsilon) \cdot n$. Recalling that at least half of the $v^{(i)}$'s are at distnace at most $2\epsilon n$ from w, it follows that these $v^{(i)}$'s are at distance at least $(\delta(\mathcal{C})/4) \cdot n$ from $\mathcal{C}$. Recall that, in case $j_2 = \ell$ (which occurs with probability $1/3$), the second subtester of T' emulates the codeword tester of $\mathcal{C}$ on a randomly chosen $v^{(i)}$. With probability at least one half, such a random $v^{(i)}$ will be at least $(\delta(\mathcal{C})/4)\cdot$ from $\mathcal{C}$, and hence the test rejects with some constant probability. The lemma follows. □

Remark 5.5 *We did not use the fact that the tester T' checks whether the list $(\ell_1, \ldots, \ell_{4m})$ is the lexicographically-first suitable one in our soundness analysis. The reason for adding these tests is aesthetic: it ensures that all non-codewords are rejected by T' with positive probability. Thus, a string is in $\mathcal{C}'$ if and only if it is accepted by T' with probability 1.*

Combining the above, we obtain:

Theorem 5.6 *For some $\delta > 0$, $s < 1$ and arbitrary large n's, there exist $[2,1,s]$-locally testable* ternary *codes $\mathcal{C}'_n \subset \{0,1,2\}^n$ with minimal relative distance δ and $|\mathcal{C}'_n| > 2^{n^{1-o(1)}}$.*

5.4 $GF(2)$-Linear Codes over $\{0,1\}^2$

The notion of $GF(2)$-linearity used here refers to viewing $\{0,1\}^2$ as a vector space (of dimension 2) over the two-element field $GF(2)$, and requires the codewords (viewed as sequences over $GF(2)$) to satisfy linear constraints over $GF(2)$. Thus, the codeword tester checks that the symbols read (viewed as 2-sequences over $GF(2)$) satisfy a linear constraint over $GF(2)$. (Indeed, this is the notion discussed in Footnote 1.)

We briefly sketch how to obtain $[2,s,1]$-locally testable codes over the four symbol alphabet $\{0,1\}^2$ that are $GF(2)$-linear, have linear distance and non-trivial rate. This improves over a corresponding result of [5] that achieves similar features for much larger alphabet (i.e., the alphabet size grows with the blocklength, n).

Our starting point is the binary code $\mathcal{C}_n$ given by Theorem 5.1, assuming a uniform tester T over $m = cn$ tests, where each test is a parity of three bits. We view this code as a code $\mathcal{C}'_n$ over $\Sigma = \{0,1\}^2$, say, by encoding each bit b by the symbol $0b$. Our new code, denoted $\mathcal{C}''_n$, is formed by taking a repetitive encoding of a word $w \in \mathcal{C}'_n$ (using $c' \gg c$ copies, as in Section 5.3) and appending to it the assignment $A_w \in (\{0,1\}^2)^m$ to the first two variables in each test. Clearly, $\mathcal{C}''_n$ is $GF(2)$-linear. Furthermore, as in Section 5.3, the distances in the new code are dominated by these in the old code and thus the code's distance and rate are as desired. We now turn to the new codeword tester: Essentially this tester checks that the first part properly encodes some binary string and that the auxiliary part matches the first part (and the parity conditions). That is, on input $((v^{(1)}, \ldots, v^{(c')}), A)$, the tester proceeds as follows:

1. With probability 1/2, it performs a random test of proper repetitive encoding of a binary string. That is, it selects $i_1, i_2 \in [c']$ and $j \in [n]$ at random, and accepts if and only if the j-th symbol of $v^{(i_1)}$ equals the j-th symbol of $v^{(i_2)}$ and this symbol is in $\{00, 01\}$.

2. With probability 1/2, it performs a random test of consistency and parity. That is, it select an original test $j \in [m]$ and an index $i \in [3]$ at random, and checks whether the j-th symbol of A matches the value of a random copy of the i-th variable of the j-th original test. Letting $\mathrm{idx}_i(j) \in [n]$ denote the index of the variable that appears as the i-th variable of the j-th original test, we consider two cases:

   (a) In case $i \in \{1,2\}$, we compare the i-th coordinate of the j-th symbol of A with the second coordinate of the $\mathrm{idx}_i(j)$-th entry in $v^{(i_2)}$, where i_2 is uniformly selected in $[c']$.

   (b) In case $i = 3$, we compare the sum of the two coordinates of the j-th symbol of A with the second coordinate of the $\mathrm{idx}_i(j)$-th entry in $v^{(i_2)}$, where again i_2 is uniformly selected in $[c']$.

Thus, the first sub-tester enforces that $(v^{(1)}, \ldots, v^{(c')})$ is close to a proper encoding of a binary string, whereas the second sub-tester enforces consistency of $(v^{(1)}, \ldots, v^{(c')})$ and A (by the first case) as well as emulates the original tester T (by the second case). Notice that all tests performed are homogeneous equalities over $GF(2)$, so we get:

Theorem 5.7 *For some $\delta > 0$, $s < 1$ and arbitrary large n's, there exists a $[2,1,s]$-locally testable $GF(2)$-linear codes $\mathcal{C}''_n \subset (\{0,1\}^2)^n$ with minimal relative distance δ and $|\mathcal{C}''_n| > 2^{n^{1-o(1)}}$.*

6 Non-Perfect Completeness for Binary Codes

In this section, we show that perfect completeness is essential for Theorem 4.1 (i.e., the upper-bound regarding non-linear binary codes admitting two-query testers).

We briefly sketch how to obtain $[2, s, c]$-locally testable codes over the binary alphabet with linear distance and non-trivial rate. Indeed, by Theorem 4.1, we can only achieve this for $c < 1$. Since this construction and the problems it faces are very similar to the ternary codes of the previous section, we only sketch the main ideas and point to the differences between the two constructions, leaving the implementation details to the interested reader.

Our starting point is the code $\mathcal{C}$ given by Theorem 5.1, assuming a uniform tester T over $m = cn$ tests, where each test is a parity of three bits. Using the 2-CNF parity gadget of [1], we replace each such parity test by a 2-CNF with 12 clauses having four auxiliary variables that are dedicated to this 2-CNF. The resulting 2-CNF gadget has the following properties:

1. There exists no assignment that satisfies all 12 clauses of the resulting 2-CNF.
2. Every satisfying assignment to the three original variables can be extended to an assignment that satisfies 11 out of the 12 clauses of the resulting 2-CNF.
3. For every assignment to the three original variables that does not satisfy the parity condition, and for every assignment of the auxiliary variables, at most 10 out of the 12 clauses (of the resulting 2-CNF) are satisfied.

Our new binary code $\mathcal{C}''$ is formed by taking a repetitive encoding of a word $w \in \mathcal{C}_n$ (using $c' \gg c$ copies, as in the previous section) and appending to it an assignment $A_w \in \{0,1\}^{4m}$ to the set of auxiliary variables. (Recall, each of the m test is allocated 4 different auxiliary variables, and A_w is an assignment to these $4m$ variables.) The latter assignment (i.e., A_w) will be the lexicographically first among all assignments that satisfy 11 out of 12 of the clauses in each resulting 2-CNF gadget. As in the ternary case, the original code will dominate distances in the new code (by virtue of sufficient repetitions). We do not bother to check that the assignment provided to the auxiliary variables is the one being lexicographically-first. (Actually, we do not know whether such a check can be implemented.) This will have the effect that some elements not in the code (yet extremely close to it) will be accepted with probability that is as high as the acceptance probability of some codewords. Although somewhat unpleasing (see Remark 5.5), this does not contradict the definition of a locally testable code. Note that, since we are dealing here with 2-CNFs (rather than 3-CNFs), it is straightforward to test (by two queries) whether a random clause is satisfiable (by the provided assignment). Thus, the new codeword test operates as follows, on input $((v^{(1)}, \ldots, v^{(c')}), A)$:

1. With probability 1/2, it performs a random test of proper repetitive encoding of a binary string. That is, it selects $i_1, i_2 \in [c']$ and $j \in [n]$ at random, and accepts if and only if the j-th bit of $v^{(i_1)}$ equals the j-th bit of $v^{(i_2)}$.
2. With probability 1/2, it performs a random test of satisfaction. That is, it selects a 2-clause $j \in [12m]$ at random, determines the indices of the two variables that appear in the j-th clause and checks whether their assignment satisfies the clause. Note that the assignment of an original variable is obtained by looking at random at one of its copies provided in $(v^{(1)}, \ldots, v^{(c')})$, whereas the assignment of an auxiliary variables is obtained from A.

Note that $\mathcal{C}''$ is binary, non-linear, and essentially preserves the distance and rate of $\mathcal{C}$. The above tester has query complexity 2 (and is non-adaptive). It can be shown that it has completeness 11/12, and soundness error $(11 - \epsilon)/12$, for some constant $\epsilon > 0$. We get:

Theorem 6.1 *For some $\delta > 0$, $s < c = 11/12$ and arbitrary large n's, there exists a $[2, c, s]$-locally testable binary codes $\mathcal{C}''_n \subset \{0,1\}^n$ with minimal relative distance δ and $|\mathcal{C}''_n| > 2^{n^{1-o(1)}}$.*

References

[1] M. Bellare, O. Goldreich and M. Sudan. Free Bits, PCPs and Non-Approximability – Towards Tight Results. *SIAM Journal on Computing*, Vol. 27, No. 3, pages 804–915, 1998.

[2] E. Ben-Sasson, P. Harsha, S. Raskhodnikova. Some 3-CNF Properties are Hard to Test. In *35th STOC*, 2003.

[3] K. Friedl and M. Sudan. Some Improvements to Total Degree Tests. In *Proc. ISTCS*, pages 190-198, 1995.

[4] O. Goldreich, S. Goldwasser, D. Ron. Property Testing and its connection to Learning and Approximation. *Journal of the ACM*, 45(4):653–750, July 1998.

[5] O. Goldreich and M. Sudan. Locally Testable Codes and PCPs of Almost-Linear Length. In *43rd FOCS*, pages 13–22, 2002.

[6] M. Kiwi. *Probabilistically Checkable Proofs and the Testing of Hadamard-like Codes.* Ph.D. Thesis, MIT, 1996.

[7] R. Rubinfeld and M. Sudan. Robust characterization of polynomials with applications to program testing. *SICOMP*, Vol. 25(2), pages 252–271, 1996.