Invertible Zero-Error Dispersers and Defective Memory with Stuck-At Errors

Ariel Gabizon
Technion

Ronen Shaltiel*
University of Haifa

June 10, 2012

Abstract

Kuznetsov and Tsybakov [11] considered the problem of storing information in a memory where some of the cells are 'stuck' at certain values. More precisely, For $0 < r, p < 1$ we want to store a string $z \in \{0,1\}^{rn}$ in an n-bit memory $x = (x_1, \ldots, x_n)$ in which a subset $S \subseteq [n]$ of size pn are stuck at certain values $u_1, \ldots, u_{pn}$ and cannot be modified. The encoding procedure receives S, $u_1, \ldots, u_{pn}$ and z and can modify the cells outside of S. The decoding procedure should be able to recover z given x (*without having to know S or $u_1, \ldots, u_{pn}$*). This problem is related to, and harder than, the Write-Once-Memory (WOM) problem (in which once we raise a cell x_i from zero to one, it is stuck at this value).

We give explicit schemes with rate $r \geq 1 - p - o(1)$ (note that $r \leq 1 - p$ is a trivial lower bound). This is the first explicit scheme with asymptotically optimal rate. We are able to guarantee the same rate even if following the encoding, the memory x is corrupted in $o(\sqrt{n})$ adversarially chosen positions. This more general setup was first considered by Tsybakov [26] (see also [10, 8]) and our scheme improves upon previous results.

Our approach utilizes a recent connection observed by Shpilka [23] between the WOM problem and linear seeded extractors for bit-fixing sources. We generalize this observation and show that memory schemes for stuck-at memory are equivalent to zero-error seedless dispersers for bit-fixing sources. We furthermore show that using zero-error seedless dispersers for affine sources (together with linear error correcting codes with large dual distance) allows the scheme to also handle adversarial errors.

It turns out that explicitness of the disperser is not sufficient for the explicitness of the memory scheme. We also need that the disperser is *efficiently invertible*, meaning that given an output z and the linear equations specifying a bit-fixing/affine source, one can efficiently produce a string x in the support of the source on which the disperser outputs z.

In order to construct our memory schemes, we give new constructions of zero-error seedless dispersers for bit-fixing sources and affine sources. These constructions improve upon previous work by [15, 6, 2, 28, 13] in that for sources with min-entropy k, they (i) achieve larger output length $m = (1 - o(1)) \cdot k$ whereas previous constructions did not, and (ii) are efficiently invertible, whereas previous constructions do not seem to be easily invertible.

*This research was supported by BSF grant 2010120, ISF grants 686/07 and 864/11, and ERC starting grant 279559.

1 Introduction

1.1 Background

Kuznetsov and Tsybakov [11] considered the problem of storing information on a defective memory with "stuck-at" errors. In this setup we have a memory $x = (x_1, \dots, x_n)$ of n cells each storing a symbol in some finite alphabet (in this paper we will restrict attention to the Boolean alphabet). The problem is that a subset $S \subseteq [n]$ containing at most s out of the n cells are 'stuck' at a certain "defect pattern" (namely, $x|_S = u$ for some $u \in \{0,1\}^{|S|}$) and we cannot modify these cells. The goal is to store a string $z \in \{0,1\}^m$ in memory x, so that at a later point it would be possible to read x and retrieve z, *even without knowing which of the cells are stuck*. Naturally, we want m to be as large as possible (as a function of n and s). A precise definition follows.

Definition 1.1 (Recovering from stuck-at errors)**.** *For positive integers $s < n$, an (n,s)-*stuck-at memory scheme *consists of*

- *a (possibly randomized) encoding function E such that given any $S \subset [n]$ with $|S| \le s$, $u \in \{0,1\}^{|S|}$ and $z \in \{0,1\}^m$, E returns $x \in \{0,1\}^n$ with $x|_S = u$, and*
- *a decoding function $D : \{0,1\}^n \mapsto \{0,1\}^m$ such that for any $x \in \{0,1\}^n$ produced by E on inputs z, S and u as above, $D(x) = z$.*

The rate *of the scheme is defined as m/n. We say that a scheme (more precisely, a sequence of schemes with $n \mapsto \infty$) is* explicit *if D is computable in deterministic poly(n)-time and E is computable in randomized expected poly(n)-time.*[1]

Motivation for this model has come recently from Phase-Change-Memory where 'stuck-at' errors are common. It may be the case that discovering the 'stuck-at' cells will be time consuming, and that the process may ruin the written content. Thus, the assumption about only the encoder knowing the defect pattern (S, u) makes sense in such a scenario. See the introduction of [12] for more details.

Connection to the standard coding theoretic setup. It is trivial that a standard error correcting code which corrects s adversarial errors can be used to solve the case of stuck-at errors. The encoding function can simply ignore the knowledge that it has of the defect pattern (S, u) and start by encoding $z \in \{0,1\}^m$ as an n bit string x using the code, and then modify $x|_S$ so that $x|_S = u$. Decoding from stuck-at errors is then simply standard decoding.

The goal is to come up with better schemes (namely obtain schemes with better rate). We remark that an advantage of using error-correcting codes for s adversarial errors is that this approach immediately extends to handle combinations of adversarial and stuck-at errors. We will consider this combined setup later on in Section 1.6.

1.2 Previous work and our results

It trivially holds that in any (n,s)-stuck-at memory scheme we have that $m \le n - s$. In fact, this holds even if the decoding procedure knows the defect pattern. For simplicity, we will often set

[1] In Definition 1.1 we allow the encoding function to be randomized. This makes sense in the application, and the solutions that we propose in this paper are indeed randomized. We stress that we do not assume that the decoding function has access to the coin tosses of the encoding function.

$s = pn$ for some constant p and measure the rate of a (family of) schemes as n grows. The trivial bound above gives that the rate of any (n, pn)-stuck-at memory scheme is at most $1-p$. Kuznetsov and Tsybakov [11] showed schemes with rate approaching $1-p$ exist by a non-constructive argument. Tsybakov [25], showed that given a linear code $C \subseteq \{0,1\}^n$ of rate r whose dual code has relative distance p, one can construct an (n, pn)-stuck-at memory scheme of rate $R = 1 - r$. Using current explicit constructions of codes, this gives a scheme of rate smaller than $1 - h(p) < 1 - p$, where h is the binary entropy function. Moreover, upper bounds on the rate of binary codes show that the method of [25] cannot give schemes with rate approaching $1 - p$, even given optimal code constructions. In this paper we construct a near optimal scheme, which comes within an additive term of $\log^{O(1)} n$ from the trivial bound.

Theorem 1.2 (Explicit scheme for defective memory with stuck-at errors)**.** *There exists a constant $c > 1$ such that for every function $s(n) \leq n - (\log n)^c$ we construct an explicit $(n, s(n))$-stuck-at memory scheme with $m(n) = n - s(n) - \log^c n$.*

In particular, when setting $s(n) = pn$ we obtain rate $1 - p - o(1)$ which is asymptotically optimal. This is the first explicit scheme with rate approaching $1 - p$.

Corollary 1.3 (Explicit asymptotically optimal scheme)**.** *For every constant $0 < p < 1$ we construct an explicit (n, pn)-stuck-at memory scheme with rate $1 - p - o(1)$.*

1.3 Connection to Write Once Memory (WOM)

Another motivation for defective memory with stuck-at errors is the setting of "Write-Once-Memory" (abbreviated as WOM) introduced by Rivest and Shamir[17]. In this setting the memory cells $x_1, \ldots, x_n$ are initialized to the value '0', and it is possible to modify a cell from '0' to '1' but not vice-versa. The goal here is to come up with schemes that allow reusing the memory x many times, where in each round we dispose the old content and want to store new content. For concreteness let us consider the simplest two round setup: We first store some string $z_1 \in \{0,1\}^{m_1}$ in memory (by encoding it as an n bit string x) and later (when we no longer need to remember z_1) we wish to reuse the memory in order to store some string $z_2 \in \{0,1\}^{m_2}$. Note that at this phase the cells containing '1' are stuck, and we need to solve an instance of the defective memory with stuck-at errors problem.

An optimal solution to the problem of stuck-at errors immediately translates into an optimal solution to the WOM problem as follows: Identify the set of strings $z_1 \in \{0,1\}^{m_1}$ with the set of strings $x \in \{0,1\}^n$ of Hamming weight pn (by choosing p such that $m_1 = h(p) \cdot n$). At the first round, we store z_1 in memory by storing the corresponding string $x \in \{0,1\}^n$ of Hamming weight pn (and note that we can indeed recover z_1 given x). This leaves us with an instance of the memory problem with pn stuck-at errors when we want to store $z_2 \in \{0,1\}^{m_2}$ in the second round. If we use a scheme with rate approaching $1 - p$, then the induced WOM-scheme has rate approaching $h(p) + 1 - p$ which is known to be optimal [17] and this matches the best known explicit schemes [23].

The WOM problem seems easier than the problem of stuck-at errors in the sense that the locations of cells that are stuck at the beginning of the second round are not arbitrary (and can be chosen by how we implement the encoding in the first round). In particular, the WOM-scheme can choose a parameter $t = o(n)$ and decide not to write in the first t cells during the first round. This allows the encoding in the second round to use these cells to pass t bits of "control information" to the decoding procedure. We remark that this approach is used in many of the WOM schemes in the literature.

This approach seems less robust to changes in the model (such as added stuck-at errors or WOM with few adversarial errors) as the decoding procedure may critically depends on correctly receiving the control information. Consequently, it seems to us that the approach of this paper (and specifically the results presented later in Section 1.6 in a setup where both stuck-at and adversarial errors occur) can lead to more robust solutions to the WOM problem.

1.4 Decoding using zero-error dispersers for bit-fixing sources

Shpilka's observation. The starting point for this work is a recent observation of Shpilka [23] which relates the problem of WOM to certain "linear seeded extractors for bit-fixing sources" (that we elaborate on in Section 2). In addition to the WOM problem, Shpilka also considers the problem of defective memory with stuck-at errors. However, his approach is not directly suitable to this problem, and instead he solves a relaxed case in which the encoding function is allowed to transfer $t = O(\log^3 n)$ bits of "control information" to the decoding function. This can be realized if both encoding and decoding procedures have access to an additional t bit external memory which is guaranteed not to have stuck-at errors.

Loosely speaking, the reason for needing an external memory is that the encoding function needs to transfer control information (which is a "seed" for the seeded extractor) so that it will be available for the decoding procedure. As explained above, in the setup of WOM, an additional external memory is not necessary because the encoding scheme can reserve the first $t = O(\log^3 n)$ cells for passing control information to the decoding procedure.

Seedless extractors and dispersers for bit-fixing sources. We would like to use Shpilka's approach while avoiding the use of external memory. This suggests that we want to replace seeded extractors with *seedless* extractors (so that no control information needs to be passed). Indeed, our first step is to recast Shpilka's observation in the terminology of "seedless zero-error dispersers for bit-fixing sources". We begin by defining seedless extractors and dispersers for a general class $\mathcal{C}$ of sources, and then define the class of bit-fixing sources.

Definition 1.4 (min-entropy and statistical distance). *Let X be a distribution over $\{0,1\}^n$. The* min-entropy *of X denoted by $H_\infty(X)$ is defined by $H_\infty(X) = min_{x\in\{0,1\}^n} \log(1/\Pr[X = x])$. Two distributions X, Y over $\{0,1\}^n$ are* ϵ-close *if for every $A \subseteq \{0,1\}^n$, $|\Pr[X \in A] - \Pr[Y \in A]| \leq \epsilon$.*

Definition 1.5 (Seedless extractors and dispersers). *Let $\mathcal{C}$ be a class of distributions over $\{0,1\}^n$. For $0 \leq k \leq n$ we use $\mathcal{C}_k$ to denote the class of distributions $X \in \mathcal{C}$ with $H_\infty(X) \geq k$.*

- *A function $E : \{0,1\}^n \mapsto \{0,1\}^m$ is an* extractor *for $\mathcal{C}$ with* entropy threshold *k and* error *$\epsilon \geq 0$ if for every $X \in \mathcal{C}_k$, $E(X)$ is ϵ-close to the uniform distribution on $\{0,1\}^m$.*
- *A function $D : \{0,1\}^n \mapsto \{0,1\}^m$ is a* disperser *for $\mathcal{C}$ with* entropy threshold *k and* error *$\epsilon \geq 0$ if for every $X \in \mathcal{C}_k$, $|\mathrm{Supp}(D(X))| \geq (1-\epsilon)2^m$ (where $\mathrm{Supp}(Z)$ denotes the support of the distribution Z). We say that D has zero-error if $\epsilon = 0$.*

We say that a (family of) extractors (or dispersers) is explicit if it runs in time poly(n).

The reader is referred to a survey article [21] for a tutorial on seedless extractors and dispersers. We will be interested in the family of bit-fixing sources.

Definition 1.6 (bit-fixing sources). *A bit-fixing source is a distribution X on $\{0,1\}^n$ such that there exists $S \subseteq [n]$ and $u \in \{0,1\}^{|S|}$ such that $X|_S$ is fixed to the value u and $X|_{[n]\setminus S}$ is uniformly distributed over $\{0,1\}^{n-|S|}$. Note that $H_\infty(X) = n - |S|$.*

Explicit memory schemes and *efficiently invertible* zero-error dispersers. We now recast Shplika's observation by noting that zero-error dispersers for bit-fixing sources with entropy threshold k that output m bits imply $(n, n-k)$-stuck-at memory schemes with rate m/n. In fact, zero-error dispersers for bit-fixing sources seem to completely capture the stuck-at problem in that the decoding procedure of any memory scheme can be shown to be a zero-error disperser. Before we state this connection, recall that our goal is to construct *explicit* schemes for stuck-at errors. Unfortunately, explicitness of the zero-error disperser is not sufficient for the induced scheme to be explicit. An additional property is needed: that the disperser is *efficiently invertible* in the sense defined below.

Definition 1.7 (Invertible zero-error dispersers). *Let $\mathcal{C}$ be a class of distributions over $\{0,1\}^n$. We say that $\mathcal{C}$ is* polynomially-specified *if each distribution $X \in \mathcal{C}$ is specified by a string of length poly(n). (For example, each bit-fixing source can be specified by the set $S \subseteq [n]$ and $u \in \{0,1\}^{|S|}$ that define the bit-fixing source).*

We say that a zero-error disperser D (for a polynomially specified class $\mathcal{C}$ with entropy threshold k) is efficiently invertible *if there is a randomized algorithm running in expected poly(n)-time that given $z \in \{0,1\}^m$ and (the specification of) a source $X \in \mathcal{C}_k$ returns $x \in \mathrm{Supp}(X)$ such that $D(x) = z$.*

We now formally state a connection between zero-error dispersers for bit-fixing sources and schemes for stuck-at errors. This connection is completely straightforward.

Theorem 1.8 (Equivalence between memory schemes and zero-error dispersers).

1. *Given a zero-error disperser $D : \{0,1\}^n \mapsto \{0,1\}^m$ for bit-fixing sources with entropy threshold k there exists an $(n, n-k)$-stuck-at memory scheme with rate m/n. Furthermore, if D is explicit and efficiently invertible then the scheme is explicit.*

2. *Given an $(n, n-k)$-stuck-at memory scheme with decoding function $D : \{0,1\}^n \mapsto \{0,1\}^m$, D is a zero-error disperser for bit-fixing sources with entropy threshold k.*

Proof. Fix a zero-error disperser $D : \{0,1\}^n \mapsto \{0,1\}^m$ for bit-fixing sources with entropy threshold k. D will be the decoding procedure of the scheme. We define the encoding procedure E as follows. Given $S \subset [n]$ with $|S| \le n-k$, $u \in \{0,1\}^{|S|}$ and $z \in \{0,1\}^m$, let X be the bit-fixing source with $\mathrm{Supp}(X) = \{x \in \{0,1\}^n : x|_S = u\}$. We have that X has min-entropy at least k. Thus, we can find $x \in \mathrm{Supp}(X)$ with $D(x) = z$. E will output such x. We indeed have $x|_S = u$ as required. Note that if D is explicit and efficiently invertible then the scheme is explicit. We move to the second item. Fix a $(n, n-k)$-stuck-at memory scheme with decoding function $D : \{0,1\}^n \mapsto \{0,1\}^m$. Let X be a bit-fixing source with min-entropy at least k and fix any $z \in \{0,1\}^m$. Then, $\mathrm{Supp}(X) = \{x \in \{0,1\}^n : x|_S = u\}$ for some $S \subset [n]$ with $|S| \le n-k$ and $u \in \{0,1\}^{|S|}$. The scheme's encoding procedure E given z, S and u, produces $x \in \mathrm{Supp}(X)$ with $D(x) = z$. Thus, D is a zero-error disperser for bit-fixing sources with entropy threshold k. □

1.5 A new construction of invertible zero-error dispersers for bit-fixing sources

By Theorem 1.8 the problem of constructing explicit $(n, s(n))$-stuck-at memory scheme is reduced to the task of explicitly constructing an efficiently invertible zero-error disperser for bit-fixing sources with entropy threshold $k = n - s(n)$. In order to prove Theorem 1.2 and achieve asymptotically optimal rate, we need dispersers with output length $m = (1 - o(1)) \cdot k$. Unfortunately, no such explicit construction is known. There are two issues:

- The best explicit construction of zero-error dispersers for bit-fixing sources was given by Gabizon and Shaltiel [6], and it only achieves output length $m = \Omega(k)$. This yields schemes with very poor rate of $\Omega(1 - p)$ which might be small even if $1 - p$ is large. (Previous constructions [3, 4, 9, 5, 15] would also give poor[2] rate when viewed as zero-error dispersers.)
- The construction of [6] is quite complicated and do not seem to be easily invertible for large values of m.

In this paper, we give an improved explicit construction of zero-error dispersers for bit-fixing sources while handling the two issues above. Namely, whenever $k >$ polylog$n$ our construction achieves $m = (1 - o(1)) \cdot k$ and is efficiently invertible.

Theorem 1.9 (Zero-error disperser for bit-fixing sources outputting almost all the bits)**.** *There exists a constant $c > 1$ such that if n is large enough and $k \geq \log^c n$, there is an explicit and efficiently invertible zero-error disperser $D : \{0,1\}^n \mapsto \{0,1\}^{k-\log^c n}$ for bit-fixing sources with entropy threshold k.*

Chor et al. [3] showed that zero-error extractors for bit-fixing sources do not exist in case $m > 1$ and $k < n/3$. In contrast, it is easy to show the existence of zero-error dispersers using the probabilistic method. Theorem 1.9 achieves output length that approaches the one given by the non-constructive argument (which gives $m = k - \log n - o(\log n)$).

Plugging this construction in Theorem 1.8 yields Theorem 1.2. We elaborate on the technique used to prove Theorem 1.9 in Section 2.

1.6 Recovering from stuck-at errors and adversarial errors

Tsybakov [26] considered a more general model of defective memory where in addition to the 'stuck-at' errors, the memory can be corrupted at few (adversarially chosen) cells after the encoding. A formal definition follows.

Definition 1.10 (Stuck-at errors and adversarial errors)**.** *An* (n, s, e)-stuck-at noisy memory scheme *consists of*

- *a (possibly randomized) encoding function E such that given any $S \subset [n]$ with $|S| \leq s$, $u \in \{0,1\}^{|S|}$ and $z \in \{0,1\}^m$, E returns $x \in \{0,1\}^n$ such that $x|_S = u$, and*
- *a decoding function $D : \{0,1\}^n \mapsto \{0,1\}^m$ such that for any $x \in \{0,1\}^n$ produced by E with input z (and any inputs S and u as above), and any 'noise vector' $\xi \in \{0,1\}^n$ of hamming weight at most e, $D(x + \xi) = z$.*

[2]An exception is the construction of Chor et. al [3] in the case of very large $k = n - o(n)$. Specifically, when $k = n - t$ [3] constructs zero-error *extractors* that output $n - O(t \cdot \log n)$ bits which is better than our Theorem 1.9 when $t = o(\log n)$. In fact, their construction, based on linear codes, is analogous to the scheme of [25] which is superior to our Theorem 1.2 when, for example, $s(n) = o(\log n)$.

The rate *of the scheme is defined as m/n. We say a scheme (more precisely, a sequence of schemes with $n \mapsto \infty$) is* explicit *if D is computable in deterministic poly(n)-time and E is computable in randomized expected poly(n)-time.*

Note that by the discussion in Section 1.1 an error-correcting code that corrects $s+e$ adversarial errors can be used to solve this more general problem. This solution seems very expensive in case $e \ll s$ (as it treats the s stuck-at errors as adversarial) and it is possible to do better in this range.

1.7 Previous work and our results

The solutions proposed in previous work (as well as our results) reduce the problem of defective memory with stuck-at and worst-case errors to constructing error-correcting codes that correct e adversarial errors. However, in all known schemes (including ours), it is required that the error correcting code has additional properties: It should be linear, and have dual distance s (meaning that the dual code should have distance at least s). We elaborate on why dual distance naturally comes up in the next section.

Let $e(\cdot), s(\cdot)$ be some integer functions and let $0 \le r_{e,s} \le 1$ denote the largest positive number such that there is an explicit family of linear binary codes with block length $n \to \infty$ such that: (i) the code corrects $e(n)$ adversarial errors (and in particular has distance at least $2e(n)+1$), (ii) the dual code has distance $s(n)$, and (iii) the rate of the code approaches $r_{e,s}$ as $n \to \infty$. (Here, by explicit family we mean that the code has encoding and decoding algorithms that run in poly(n)-time).

Kuzentsov, Kasami and Yamamura [10] proposed an $(n, s(n), e(n))$-stuck-at noisy memory scheme with rate $r_{e,s} - s(n)/n - o(1)$. However, their construction has a non-constructive component. Later, Heegard [8] made this component explicit by using partitioned linear block codes. Using current explicit constructions of binary codes, and setting $s(n) = pn$ for a constant p, an explicit $(n, pn, e(n))$-stuck-at noisy memory scheme using [8] would give rate smaller than $r_{e,pn} - h(p)$. Later work focused on improving the efficiency of the encoding and decoding procedures of [8], but not the rate [12]. In this work we give explicit schemes matching the rate guaranteed by the non-explicit argument of [10].

Theorem 1.11 (Explicit scheme that also handles adversarial errors)**.** *Let $e(\cdot), s(\cdot)$ be integer functions. For sufficiently large n, we construct an explicit $(n, s(n), e(n))$-stuck-at noisy memory scheme with rate $r_{e,s} - \frac{s(n)}{n} - o(1)$.*

It may seem restricting that in addition to correcting e adversarial errors, the code needs to be linear and have large dual distance. Nevertheless, in some cases these additional properties come at no extra cost (when measuring the rate as a function of the number of adversarial errors). One such example is the Hamming code which is an explicit linear code with distance 3 that has best possible rate amongst all codes with such distance. The dual code (which is the Hadamard code) has distance $s(n) = n/2$. Altogether, this gives that for $p \le 1/2$ and $s(n) \le pn$, we get a scheme with rate $1 - p - o(1)$ that corrects $e(n) = 1$ adversarial errors.

We can do even better and allow $e(n) = o(\sqrt{n})$ adversarial errors for the same rate of $1-p-o(1)$ by using BCH codes. For any $e(n) = o(\sqrt{n})$, BCH codes give us an explicit family of linear codes with block length $n \mapsto \infty$ that have distance $2e(n)+1$ (which in turns allows correcting $e(n)$ adversarial errors) and dimension at least $n - \log n \cdot e(n) - 1$. The dual distance of this code is at least $n/2 - e(n) \cdot \sqrt{n}$. This translates into the following corollary.

Corollary 1.12. *Let $p < 1/2$ be a constant and let $e(n) = o(\sqrt{n})$ and $s(n) \le pn$. For sufficiently large n, we construct an explicit $(n, s(n), e(n))$-stuck-at noisy memory scheme with rate $1-p-o(1)$.*

This means that we can allow $e(n) = o(\sqrt{n})$ adversarial errors at the same rate given in Corollary 1.3 (except for the identity of the function hidden in the $o(1)$ term). Furthermore, as $n \to \infty$ this rate matches the trivial bound of $1 - p$ (and recall that this bound holds even without adversarial errors and when the decoding procedure knows the defect pattern).[3]

1.8 Decoding using zero-error dispersers for affine sources

Loosely speaking, the reason that dual distance comes up naturally in the results above is that a linear code $\mathcal{C} \subseteq \mathbb{F}_2^n$ with dual distance s has the property that for every $S \subseteq [n]$ of size s and every $u \in \{0,1\}^s$, $\mathcal{C}$ has a non-empty subset $\mathcal{C}_{S,u}$ of codewords x satisfying $x|_S = u$. This follows as the $\text{rate}(\mathcal{C}) \cdot n \times n$ generator matrix of such a code has the property that every s columns are linearly independent. Once we know that $\mathcal{C}_{S,u}$ is not empty, it follows by linearity that it is in fact quite large, as it forms an affine subspace of $\mathbb{F}_2^n$ with dimension at least $\text{rate}(\mathcal{C}) \cdot n - s$.

This suggests that given (S, u), it might be a good idea that the encoding procedure of the memory scheme encodes strings $z \in \{0,1\}^m$ by strings $x \in \mathcal{C}_{S,u}$. Such strings are consistent with the defect pattern, and they form an error correcting code (that can correct as many errors as $\mathcal{C}$ can). The advantage of this approach is that it is easy for the decoding procedure of the memory scheme to handle the adversarial errors: Upon receiving a corrupted string x' (that was derived from some $x \in \mathcal{C}_{S,u}$ by e adversarial errors) one can run the decoding algorithm of $\mathcal{C}$ to recover x. At this point, the decoding procedure of the memory scheme (which does not know S, u) needs to recover the original string z by applying some polynomial time function $f : \{0,1\}^n \to \{0,1\}^m$ on x. We would like f to have to have the following property: For every $z \in \{0,1\}^m$ and every defect pattern (S, u), there exists an $x \in \mathcal{C}_{S,u}$ such that $f(x) = z$, and furthermore that such an x can be found efficiently given S, u and z. This implies that we can use efficiently invertible zero-error dispersers for affine sources with entropy threshold $\text{rate}(\mathcal{C}) \cdot n - s$ (that we define next).

Definition 1.13 (Affine sources). *An affine source X is a distribution over $\{0,1\}^n$ (identified with $\mathbb{F}_2^n$) that is uniform over some affine subspace of $\mathbb{F}_2^n$.*

Note that every bit-fixing source is also an affine source, and that the class of affine sources is polynomially specified as every affine subspace can be specified by at most n affine constraints. The argument explained above gives the following theorem.

Theorem 1.14. *Given integers s, e and n, let $\mathcal{C} \subseteq \{0,1\}^n$ be a linear code that corrects e errors and has dual distance s. Given a zero-error disperser $f : \{0,1\}^n \to \{0,1\}^m$ for affine sources with entropy threshold $\text{rate}(\mathcal{C}) \cdot n - s$, there exists an (n, s, e)-stuck-at noisy memory scheme with rate m/n. Furthermore, if $\mathcal{C}$ has polynomial time encoding and decoding, and f is explicit and efficiently invertible, then the scheme is explicit.*

[3]The function hidden in the $o(1)$ term in Corollary 1.12 is $O(1/\sqrt{\log\log n})$. This is much larger than the $o(1)$ term in Corollary 1.3 which is $O(\log^{O(1)} n/n)$. We also mention that for our choice of parameters, $r_{e(n),pn} = 1 - O(e(n) \cdot \log n/n)$. We remark that our approach can potentially achieve rate approximately $r_{e(n),pn} - pn$, and the missing component is an improved explicit construction of zero-error dispersers for affine sources (which we elaborate on in the next section). More details on potential improvements are given in Section 5.

1.9 A new construction of invertible zero-error dispersers for affine sources

By Theorem 1.14 the problem of constructing explicit $(n, s(n), e(n))$-stuck-at noisy memory scheme is reduced to the task of explicitly constructing an efficiently invertible zero-error disperser for affine sources with entropy threshold $k = r_{s,e} \cdot n - s$. Once again, we require dispersers with output length $m = (1 - o(1)) \cdot k$. Similar to the situation for bit-fixing sources, no such explicit constructions are known. Moreover, for affine sources, there are no explicit constructions with $m = \omega(1)$ for $k = o(n/\sqrt{\log \log n})$. For $k = \Omega(n/\sqrt{\log \log n})$ there is an explicit construction that achieves $m = \Omega(k)$. This is due to by Bourgain [2] with improvements by Yehudayoff [28] and Li [13] (in fact, this construction gives an extractor which is a stronger object than a disperser). For smaller k, the best known constructions by Kopparty and Ben-Sasson [1] (which handles $k = \Omega(n^{4/5})$ and Shaltiel [20] (which handles $k \geq 2^{\log^{0.9} n}$) only achieve $m = 1$. Furthermore, as was the case for bit-fixing sources, these constructions do not seem to be easily invertible.

In this paper, we give an improved construction of zero-error dispersers for affine sources. Our construction (which uses the construction of [2, 13, 28]) achieves $m = (1 - o(1)) \cdot k$ and is efficiently invertible. Plugging this construction in Theorem 1.14 yields Theorem 1.11.

Theorem 1.15 (Zero-error disperser for affine sources outputting almost all the bits)**.** *There exists a constant $\beta > 0$ such that if n is large enough and $k \geq \frac{\beta n}{\sqrt{\log \log n}}$, there is an explicit and efficiently invertible zero-error disperser $D : \{0,1\}^n \mapsto \{0,1\}^{k - \frac{\beta n}{\sqrt{\log \log n}}}$ for affine sources with entropy threshold k.*

Note that in particular, this gives $m = (1 - o(1)) \cdot k$ for linear k. We stress that that Theorem 1.15 is incomparable to the results of [2, 13, 28]. It achieves better output length, but we only obtain a zero-error disperser and not an extractor. Obviously, the disperser of Theorem 1.15 is also good for bit-fixing sources. However, it is incomparable to Theorem 1.9 as it only works for entropy threshold $k = \Omega(n/\sqrt{\log \log n})$ whereas Theorem 1.9 allows entropy threshold $k = (\log n)^{O(1)}$. Moreover, the output length of Theorem 1.9 is superior even for large k. The inferior parameters of our zero-error disperser for affine sources (compared to the case of bit-fixing sources) is the cause for the less tight bounds that we obtain on schemes in the noisy case, as discussed in a footnote in the end of the previous section.

1.10 Organization of the paper

In Section 2 we give a high level overview to the tools and techniques used to prove Theorems 1.9 and 1.15. In Section 3.1 we show how to compose a subsource hitter with short seed with a subsource hitter with large output length in order to obtain a subsource hitter with both properties. In Section 3.2 we show how to convert a subsource hitter for affine sources into a subsource hitter for bit-fixing sources. Finally, in Section 4 we implement the plan explained in Section 2 and prove Theorems 1.9 and Theorem 1.15.

2 Technique

In this section we outline the main ideas used to explicitly construct the zero-error dispersers of Theorem 1.9 and Theorem 1.15.

2.1 Explicit extractors with small output length are efficiently invertible

Our goal is to explicitly construct efficiently invertible zero-error dispersers for bit-fixing sources and affine sources. As mentioned earlier, the known constructions of zero-error dispersers do not seem to be easily invertible for large values of m. Nevertheless, for both classes of sources, explicit extractors with short output length $m = O(\log n)$ and error $\epsilon \leq 2^{-(m+1)}$ are easily seen to be efficiently invertible zero-error dispersers.

Proposition 2.1. *Let $\mathcal{C}$ be a polynomially specified class of distributions over $\{0,1\}^n$ such that given the specification of an $X \in \mathcal{C}$ there is a polynomial time randomized algorithm that samples an element according to X. Let c be a constant and set $m(n) = c \log n$. Let $E : \{0,1\}^n \to \{0,1\}^{m(n)}$ be an explicit (family of) $2^{-(m(n)+1)}$-extractors for $\mathcal{C}$ with entropy threshold $k(n)$, then E is an explicit and efficiently invertible (family of) zero-error dispersers for $\mathcal{C}$ with entropy threshold $k(n)$.*

Proof. Fix some integer n and let $m = m(n)$ and $k = k(n)$. Given $z \in \{0,1\}^m$ and a specification of a source $X \in \mathcal{C}_k$, we have that $\Pr[E(X) = z] \geq 2^{-m} - 2^{-(m+1)} \geq 2^{-(m+1)} = 1/2n^c$, and therefore if we sample x from X and apply $E(x)$ we have probability at least $1/2n^c$ to get $E(x) = z$. Therefore in expected time polynomial in n we can produce x such that $E(x) = z$.[4] $\square$

Note that both bit-fixing sources and affine sources are polynomially specified and given the specification of a source X it is easy to efficiently sample from X. Furthermore, for both classes there are known explicit extractors that output a logarithmic amount of bits with sufficiently low error. For bit-fixing sources we have extractors with entropy threshold $k = (\log n)^{O(1)}$ [15], and for affine sources we have extractors with entropy threshold $k = \Omega(n/\sqrt{\log \log n})$ [2, 28, 13].

2.2 Composition increases output length

We would like a method to take a zero-error disperser $D' : \{0,1\}^n \to \{0,1\}^{m_0}$ with small $m_0 = O(\log n)$ and increase the output length to $m = (1-o(1)) \cdot k$. A natural approach that was suggested by Gabizon and Shaltiel [6] (see also [5, 19]) is to take some function $F : \{0,1\}^n \times \{0,1\}^{m_0} \to \{0,1\}^m$ and consider the function $D : \{0,1\}^n \to \{0,1\}^m$ defined by $D(x) = F(x, D'(x))$.

Gabizon and Shaltiel tailored the following definition of "subsource-hitters" so that this composition yields zero-error dispersers. We would like D to be efficiently invertible and it turns out that this easily follows if we add the requirement that the subsource-hitter is efficiently invertible in the sense below.

Definition 2.2 (Subsource hitter [6])**.** *A distribution X' over $\{0,1\}^n$ is a* subsource *of a distribution X over $\{0,1\}^n$ if there exists $\alpha > 0$ and a distribution X'' over $\{0,1\}^n$ such that $X = \alpha \cdot X' + (1-\alpha) \cdot X''$.*

Let $\mathcal{C}$ be a polynomially specified class of distributions over $\{0,1\}^n$. A function $F : \{0,1\}^n \times \{0,1\}^t \mapsto \{0,1\}^m$ is a subsource-hitter *for $\mathcal{C}$ with* entropy threshold k *and* subsource entropy $k - v$ *if for any $X \in \mathcal{C}_k$ and $z \in \{0,1\}^m$ there exists a $y \in \{0,1\}^t$ and a distribution $X' \in \mathcal{C}_{k-v}$ that is a subsource of X such that for every $x \in \mathrm{Supp}(X')$ we have that $F(x,y) = z$.*

We say that F is efficiently invertible *if given $z \in \{0,1\}^m$ and a specification of $X \in \mathcal{C}_k$, such $y \in \{0,1\}^d$ and a specification of $X' \in \mathcal{C}_{k-v}$ can be found in expected poly(n)-time.*

[4]We remark that the proof works also if E is not an extractor and instead satisfies the weaker property that for for every $X \in \mathcal{C}_k$, and every $z \in \{0,1\}^m$, $\Pr[E(X) = z] \geq 1/n^{O(1)}$.

Theorem 2.3 (Composition theorem [6]). *Let $\mathcal{C}$ be a polynomially specified class of distributions over $\{0,1\}^n$. Given*

- *a zero-error disperser $D' : \{0,1\}^n \mapsto \{0,1\}^{m_0}$ for $\mathcal{C}$ with entropy threshold $k - v$, and*
- *a subsource hitter $F : \{0,1\}^n \times \{0,1\}^{m_0} \mapsto \{0,1\}^m$ for $\mathcal{C}$ for entropy threshold k and subsource entropy $k - v$,*

the function $D(x) \triangleq F(x, D'(x))$ is a zero-error disperser for $\mathcal{C}$ with entropy threshold k. Furthermore, if F and D' are explicit and efficiently invertible, then so is D.

Proof. Given $z \in \{0,1\}^m$, as F is efficiently invertible we can efficiently find a specification of X' and y as in Definition 2.2. Now, as D' is efficiently invertible we can efficiently find $x \in \mathrm{Supp}(X')$ such that $D'(x) = y$. It follows, that $D(x) = F(x, D'(x)) = F(x, y) = z$. □

Summing up, in order to construct our dispersers, it suffices to design explicit and efficiently invertible subsource-hitters for bit-fixing sources and affine sources with appropriate parameters.[5]

2.3 Linear-seeded extractors and subsource-hitters for affine sources

Seeded extractors [14] are functions that use a short seed of few truly random bits to extract many random bits from arbitrary distributions with sufficiently large min-entropy. Seeded extractors have many applications in TCS. The reader is referred to the survey articles [18, 21, 27]. We will make use of *strong linear seeded extractors* defined below.

Definition 2.4 (Strong linear seeded extractors). *A function $E : \{0,1\}^n \times \{0,1\}^d \mapsto \{0,1\}^m$ is a strong (k, ϵ)-linear seeded extractor if*

- *for every distribution X of min-entropy k, for a $1 - \epsilon$ fraction of $y \in \{0,1\}^d$, the distribution $E(X, y)$ is ϵ-close to uniform.*
- *For every $y \in \{0,1\}^d$, the function $f_y(x) = E(x, y)$ is an $\mathbb{F}_2$-linear function of x.*

We say that (a family of) extractors E is explicit if there is an algorithm that given $y \in \{0,1\}^d$ produces a matrix realizing the linear function $f_y(x)$ in poly(n)-time .

An easy observation is that explicit strong linear seeded extractors are in particular explicit and efficiently invertible subsource-hitters for affine sources.

Proposition 2.5. *Suppose $E : \{0,1\}^n \times \{0,1\}^d \mapsto \{0,1\}^m$ is an explicit strong $(k, 1/4)$-linear seeded extractor, then it is an explicit and efficiently invertible subsource-hitter for affine sources with entropy threshold k and subsource entropy $k - m$.*

[5]We will not elaborate on the precise choice of parameters in this high level overview. Note however, that a difficulty in the composition method is that when we shoot for entropy threshold k, we require that the "initial disperser" D' has entropy threshold $k - v$. It is easy to show that v has to satisfy $v \geq m$ where m is the output length of the subsource hitter. Thus, if we want to output m bits for m close to k, then we need the initial disperser D' to have very small entropy threshold. Recall that in the case of bit-fixing sources, we have initial dispersers with much smaller entropy threshold than in the case of affine sources, and this is why we obtain better results for bit-fixing sources even when shooting for large entropy threshold k.

Proof. For every affine source X of min-entropy $\geq k$, we have that for a $3/4$-fraction of $y \in \{0,1\}^d$, the distribution $E(X,y)$ is $1/4$-close to uniform. For such y, by the linearity requirement, $E(X,y)$ is an affine source, and therefore must be completely uniform and have full support. Thus, for every $z \in \{0,1\}^m$ the set $G = \{x \in \mathrm{Supp}(X) : E(x,y) = z\}$ forms an affine subspace of dimension $k-m$ and the uniform distribution X' over G is a subsource with min-entropy $k-m$ on which $E(\cdot,y) = z$. Efficient invertibility follows as if we sample $y \in \{0,1\}^d$, we have probability $3/4$ to obtain a good y, and once we obtain one we can identify that it is good by solving the linear equations defining G. These equations also give us the specification of X' as required. $\square$

2.4 Subsource-hitters for affine sources

We now turn our attention to constructing the zero-error disperser for affine sources of Theorem 1.15. By the previous discussion, an explicit strong linear seeded $(k,1/4)$-extractor $F : \{0,1\}^n \times \{0,1\}^{O(\log n)} \to \{0,1\}^m$ with $m = (1-o(1)) \cdot k$ can be composed with the extractor of [2, 28, 13] for affine sources to yield the desired zero-error disperser. Unfortunately, no such explicit construction of a strong linear seeded extractor is known. The best known explicit constructions of strong linear seeded extractor with $d = O(\log n)$ [24, 22] achieve output length $k^\alpha \leq m \leq k^{1-\alpha}$ for some constant $\alpha > 0$ (depending on the range of k). On the other hand, if we insist on output length $m = (1-o(1)) \cdot k$, then the best known explicit construction [16] requires seed length $d = \Theta(\log^3 n)$.

A natural approach is to compose these two linear seeded extractors. Namely, let $F_1 : \{0,1\}^n \times \{0,1\}^{O(\log n)} \to \{0,1\}^{O(\log^3 n)}$ be the strong linear seeded extractor of [24, 22] and $F_2 : \{0,1\}^n \times \{0,1\}^{O(\log^3 n)} \to \{0,1\}^{(1-o(1)) \cdot k}$ be the strong linear seeded extractor of [16], and consider $F : \{0,1\}^n \times \{0,1\}^{O(\log n)} \to \{0,1\}^{(1-o(1)) \cdot k}$ defined by $F(x,y) = F_2(x, F_1(x,y))$. While this is not necessarily a strong linear seeded extractor, it can be easily shown to be an efficiently invertible subsource-hitter for affine sources. In Section 3.1 we state and prove a general result showing that for any family of polynomially specified sources, the composition of two efficiently invertible subsource-hitters (with appropriate parameters) is an efficiently invertible subsource-hitter.

Summing up, we obtain the disperser guaranteed in Theorem 1.15 by $D(x) = F_2(x, F_1(x, D'(x)))$ where D' is the extractor of [2, 28, 13], and F_1, F_2 are the strong linear seeded extractor of [24, 22] and [16]respectively. The precise details are given in Section 4.3.

2.5 Subsource-hitters for bit-fixing sources

We now discuss the proof of Theorem 1.9 and the case of bit-fixing sources. We would like to follow the same outline used for affine sources. To implement this plan we need to explicitly construct efficiently invertible subsource-hitters for bit-fixing sources. Unfortunately, the best known construction of subsource-hitters for bit-fixing sources [6] does not seem to be easily invertible, and can at best achieve $m = \Omega(k)$ (and this is why the zero-error disperser of [6] only achieved $m = \Omega(k)$). In this paper, we give a different construction that is efficiently invertible and achieves $m = (1-o(1)) \cdot k$. For this purpose we show how to transform a given subsource-hitter for affine sources into a subsource-hitter for bit-fixing sources. The target subsource-hitter has seed that is increased by an additive factor of $O(\log n)$ (and slightly damaged entropy threshold and subsource entropy). The precise statement and construction appear in Section 3.2.

Using the transformation above we can transform the previously constructed subsource-hitter for affine sources into a subsource-hitter for bit-fixing sources. The final zero-error disperser is then obtained by using Theorem 2.3 to compose this subsource-hitter with Rao's extractor for bit-fixing sources (which by Proposition 2.1 is an efficiently invertible zero-error disperser for bit-fixing sources with suitable parameters). The precise details appear in Section 4.2.

3 Composing subsource hitters

3.1 Composing to get both short seed and large output

We show that a composition of two subsource-hitters yields a subsource-hitter inheriting the seed of the first one and the output length of the second one.

Proposition 3.1 (Composition of subsource-hitters). *Let $\mathcal{C}$ be a polynomially specified family of distributions over $\{0,1\}^n$. Given,*

- *A subsource-hitter $F_1 : \{0,1\}^n \times \{0,1\}^{d_1} \to \{0,1\}^{d_2}$ for $\mathcal{C}$ with entropy threshold $k - v_2$ and subsource entropy $k - v_1 - v_2$, and*
- *A subsource-hitter $F_2 : \{0,1\}^n \times \{0,1\}^{d_2} \to \{0,1\}^m$ for $\mathcal{C}$ with entropy threshold k and subsource entropy $k - v_2$.*

Then the $F : \{0,1\}^n \times \{0,1\}^{d_1} \to \{0,1\}^m$ defined by $F(x,y) = F_2(x, F_1(x,y))$ is a subsource-hitter for $\mathcal{C}$ with entropy threshold k and subsource entropy $k - v_1 - v_2$. Furthermore, if F_1, F_2 are explicit and efficiently invertible then so is F.

Proof. Fix $z \in \{0,1\}^m$ and a distribution $X \in \mathcal{C}_k$. We will prove the furthermore part, and will show how to efficiently produce $y \in \{0,1\}^{d_1}$ and the specification of $X' \in \mathcal{C}_{k-v_1-v_2}$ such that for every $x \in \mathrm{Supp}(X')$, $F(x,y) = z$. We apply the inverting algorithm of F_2 on z and X and obtain in expected poly(n)-time a string $y_2 \in \{0,1\}^{d_2}$ and the specification of a subsource $X^{(2)} \in \mathcal{C}_{k-v_2}$ of X such that for every $x \in \mathrm{Supp}(X^{(2)})$, $F_2(x, y_2) = z$. We now apply the inverting algorithm of F_1 on y_2 and the specification of $X^{(2)}$ and obtain in expected poly(n)-time, a string $y \in \{0,1\}^{d_1}$ and the specification of a subsource $X' \in \mathcal{C}_{k-v_1-v_2}$ of $X^{(2)}$ such that for every $x \in \mathrm{Supp}(X')$, $F_1(x,y) = y_2$. Thus, we have that for every $x \in \mathrm{Supp}(X')$,

$$F(x,y) = F_2(x, F_1(x,y)) = F_2(x, y_2) = z.$$

□

3.2 Subsource hitters: from affine sources to bit-fixing sources

In this section we show how to transform a subsource-hitter for affine sources into a subsource-hitter for bit-fixing sources (without spoiling the parameters by much). The precise statement is given below and is tailored for the application in Section 4.3.

Lemma 3.2. *Let $c > 1$ be some constant. Let $F' : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ be an explicit and efficiently invertible subsource-hitter for affine sources with entropy threshold $k - 6\log^c n$ and positive subsource entropy. We construct an explicit and efficiently invertible subsource-hitter $F : \{0,1\}^n \times \{0,1\}^{d+O(\log k)} \to \{0,1\}^m$ for bit-fixing sources with entropy threshold k and subsource entropy $\log^c n$.*

The remainder of this section is devoted to proving Lemma 3.2 and will rely on constructions of averaging samplers. It will be convenient to use the following non-standard definition of a sampler. The reader is referred to [7] for a survey article on samplers.

Definition 3.3. *An $(n, k, k_{min}, k_{max}, \delta)$-sampler $Samp : \{0,1\}^t \to P([n])$ is a function such that for any $S \subseteq [n]$ such that $|S| = k$:*

$$\Pr_{w \leftarrow U_t}(k_{min} \le |Samp(w) \cap S| \le k_{max}) \ge 1 - \delta$$

The sampler is explicit if it can be computed in poly(n)-time.

We use the following explicit construction of samplers from [5].

Theorem 3.4. *[5] For sufficiently large n, and every $r \le k \le n$, and $\epsilon \ge 1/kr$ there is an explicit $(n, k, k/2r, 3k/r, \epsilon)$-sampler with $t = O(\log r + \log\log n + \log(1/\epsilon))$.*

In particular, we get the following corollary by setting $r = k/2\log^c n$ and $\epsilon = 1/k$.

Corollary 3.5. *For every constant $c > 1$, sufficiently large n, and $2\log^c n \le k \le n$ there is an explicit $(n, k, \log^c n, 6\log^c n, 1/k)$-sampler with $t = O(\log k)$.*

We are now ready to prove Lemma 3.2.

Proof. (of Lemma 3.2) Let $c > 1$ be some constant, and let F' be the subsource-hitter for affine sources guaranteed in Lemma 3.2. We construct $F : \{0,1\}^n \times \{0,1\}^{d+O(\log k)} \to \{0,1\}^m$ as follows: F receives inputs $x \in \{0,1\}^n$ and $y \in \{0,1\}^{d+O(\log k)}$. We think of y as a pair $y = (y_1, y_2)$ where $y_1 \in \{0,1\}^d$ and $y_2 \in \{0,1\}^{O(\log k)}$ is of length suitable to apply the sampler of Corollary 3.5. The procedure $F(x, y)$ first applies $Samp(y_2)$ to obtain a set $T \subseteq \{0,1\}^n$. The final output of F is $F'(x|_{[n]\setminus T}, y_1)$. (Formally, we need to pad $x|_{[n]\setminus T}$ with zeros so that it becomes an n-bit string).

We now show that F is indeed an efficiently invertible subsource-hitter for bit-fixing sources with entropy threshold k and subsource entropy $\log^c n$. Let X be a bit-fixing source with min-entropy k. That is, there exists a set S of size k such that $X|_S$ is uniform and $X|_{[n]\setminus S}$ is fixed. Given the specification of X and some $z \in \{0,1\}^m$, we need to efficiently produce a specification of a bit-fixing source X' that is a subsource of X, and $y = (y_1, y_2)$ such that for every $x \in \mathrm{Supp}(X')$, $F(x, (y_1, y_2) = z)$. We do this as follows. We first go over all $k^{O(1)}$ $y_2 \in \{0,1\}^{O(\log k)}$ until we find a string y_2 such that $T = Samp(y_2)$ satisfies $\log^c n \le |T \cap S| \le 6\log^c n$. Such a string y_2 exists by Corollary 3.5. Given z and the specification of X, we consider the distribution $X|_{[n]\setminus T}$ (padded with zeros to obtain length n). This is a bit-fixing source with min-entropy $|S \setminus (S \cap T)| = |S| - |S \cap T| \ge k - 6\log^c n$. It is therefore also an affine source. We compute its specification (as an affine source) and apply the inverting algorithm of the subsource-hitter F' on z and this specification. In expected polynomial time we obtain $y_2 \in \{0,1\}^d$ and the specification of an affine source W over $\{0,1\}^{[n]\setminus T}$ such that for every $w \in \mathrm{Supp}(W)$, applying F' on w (padded to length n) and y_2 gives z. Note that we can efficiently obtain $w \in \mathrm{Supp}(W)$ by solving a set of linear equations. Fix such a w. We consider the distribution $X' = (X|X|_{[n]\setminus T} = w)$. This is a subsource of X which is a bit-fixing source with min-entropy $|S \cap T| \ge \log^c n$. Furthermore, for every $x \in \mathrm{Supp}(X')$ we have that

$$F(x, (y_1, y_2)) = F'(x|_{[n]\setminus T}, y_1) = F'(w, y_1) = z.$$

$\square$

4 Explicit constructions of zero-error dispersers

In this section we implement the plan outlined in Section 2 and prove Theorems 1.9 and 1.15.

4.1 Existing constructions of strong linear seeded extractors

We will make use of several explicit constructions of strong linear seeded extractors. We list these below starting with the seminal work of Trevisan [24].

Theorem 4.1 (Trevisan's extractor [24]). *There exists a constant $\alpha > 0$ such that for every sufficiently large n and k there is an explicit strong linear seeded $(k, 1/4)$-extractor $E_{Trevisan} : \{0,1\}^n \times \{0,1\}^{O(\frac{\log^2 n}{\log k})} \to \{0,1\}^{k^{1-\alpha}}$.*

Trevisan's extractor achieves seed length $O(\log n)$ if $k \geq n^\delta$ for some fixed $\delta > 0$. A subsequent construction of [22] achieves seed length $O(\log n)$ for every k.

Theorem 4.2 (Shaltiel-Umans extractor [22]). *There exists a constant $0 < \alpha < 1/4$ such that for every sufficiently large n and k there is an explicit strong linear seeded $(k, 1/4)$-extractor $E_{SU} : \{0,1\}^n \times \{0,1\}^{O(\log n)} \to \{0,1\}^{k^{1-\alpha}}$.*

Another construction subsequent to Trevisan is due to Raz, Reingold and Vadhan [16], and is able to extract $m = k - O(1)$ bits using seeds of length $O(\log^3 n)$.

Theorem 4.3 (Raz, Reingold and Vadhan extractor [16]). *For every sufficiently large n and $k \leq n$ there is an explicit strong linear seeded $(k, 1/4)$-extractor $E_{RRV} : \{0,1\}^n \times \{0,1\}^{O(\log^3 n)} \to \{0,1\}^{k-O(1)}$.*

Finally, it immediately follows that if we trim the output of a strong linear seeded extractor from m to $m - a$ bits by removing the last a bits, one obtains a strong linear seeded extractor for shorter output length. Therefore, we will allow ourselves to apply these extractors with shorter output lengths than the ones specified in the theorems.

4.2 Proof of Theorem 1.15

In this section we prove Theorem 1.15. We consider the explicit strong linear seeded extractors E_{SU} of Theorem 4.1 and E_{RRV} of Theorem 4.3. By proposition 2.5 each one of these is also an explicit and efficiently invertible subsource-hitter for affine sources where the entropy threshold k is inherited from the extractor and the subsource entropy is $k - m$ where m is the output length of the extractor. Let $k \geq \log^4 n$ (so that the output of E_{SU} is of length $\omega(\log^3 n)$). Let $d_{RRV} = O(\log^3 n)$ be the seed length of E_{RRV}. We choose the output lengths so that $m_{SU} = d_{RRV}$. Note that we can choose m_{RRV} to be any integer smaller than $k - a$ for some constant $a > 0$. We can now compose these two subsource-hitters using Proposition 3.1. We obtain the following result.

Corollary 4.4. *There exists a constant $c > 1$ such that for every sufficiently large n, $k \geq \log^4 n$ and $m \leq k - \log^4 n$ there is an explicit and efficiently invertible subsource-hitter $F : \{0,1\}^n \times \{0,1\}^{c \log n} \to \{0,1\}^m$ for affine sources with entropy threshold k and subsource entropy $k - m - O(\log^3 n)$.*

We can now use Theorem 2.3 to compose F with any zero-error disperser for affine sources E that extracts $c \log n$ bits. By proposition 2.1 we have that any explicit extractor $E : \{0,1\}^n \to \{0,1\}^{c \log n}$ for affine sources with entropy threshold k' and sufficiently small error is a suitable efficiently invertible and explicit zero-error disperser. We only need that to make sure that $k - m - O(\log^3 n) \geq k'$ so that we can afford the composition. We will use the extractor of [2, 13, 28] in which $k' = \Omega(n/\sqrt{\log \log n})$, and therefore, by Theorem 2.3 we can extract $m = k - k' - O(\log^3 n)$ bits for every k for which m defined above is positive. Altogether, this proves Theorem 1.15.

4.3 Proof of Theorem 1.9

In this section we prove Theorem 1.9. We consider the explicit and efficiently invertible subsource-hitter for affine sources of Corollary 4.4. We can apply Lemma 3.2 to transform this subsource-hitter into a subsource-hitter for bit-fixing sources. We obtain the following Corollary.

Corollary 4.5. *There exists a constant c such that for every constant $c' > 1$ and for every sufficiently large n, $k \geq \log^{c'} n$ and $m = k - O(\log^{c'} n)$ there is an explicit and efficiently invertible subsource-hitter $F : \{0,1\}^n \times \{0,1\}^{c \log n} \to \{0,1\}^m$ for bit-fixing sources with entropy threshold k and subsource entropy $\log^{c'} n$.*

Rao [15] gave an explicit construction of extractors for bit-fixing sources. His construction gives that there exists a constant $c' > 1$ such that for every $k \geq \log^{c'} n$ it is possible to extract more than $\sqrt{k}$ bits with error $2^{-k^{\Omega(1)}}$. In particular we can choose output length $m = c \log n$ for any constant c and error $\epsilon = 2^{-(m+1)}$. Applying Proposition 2.1, we obtain an explicit and efficiently invertible zero-error disperser for entropy threshold $k = \log^{c'} n$. The disperser of Theorem 1.9 is obtained by applying the composition of Theorem 2.3 on the subsource-hitter of Corollary 4.5 and this zero-error disperser.

5 Conclusion and Open problems

In this paper we give explicit constructions of:

- Schemes with asymptotically optimal rate for defective memory with stuck-at errors and possibly few adversarial errors.
- Improved zero-error dispersers for bit-fixing sources and affine sources.

An interesting open problem is to improve the output length of our zero-error dispersers for affine sources. (Note that we currently get output length $m = k - \log^{O(1)} n$ for bit-fixing sources, and only $m = k - O(n/\sqrt{\log \log n})$ for affine sources).

Getting improvements in the case of affine sources will allow us to improve the bounds we get on $m(n)$ in the case where there are both stuck-at errors and adversarial errors. This matters especially in settings where $r_{e,s} = 1$. More specifically, for codes $\mathcal{C}$ with dimension $n - a(n)$ for $a(n) = o(n)$ (such as the Hamming and BCH codes that we use in our schemes), matching the output length we obtain for bit-fixing sources will allow us to show that $m(n) = n - a(n) - s(n) - \log^{O(1)} n$, whereas we currently achieve $m(n) = n - a(n) - s(n) - O(n/\sqrt{\log \log n})$.

If we plan to use our composition method to construct improved dispersers for affine sources, then we need to first solve the case of dispersers for affine sources with low entropy threshold. It suffices to output $m = \Theta(\log n)$ bits to "jump-start" our approach. Nevertheless, we remark that all known explicit constructions for small k [1, 20] achieve only $m = 1$.

It may also be interesting to try and come up with schemes where the encoding procedure is deterministic rather than randomized.

Acknowledgements

We thank Amir Shpilka for many helpful conversations. We thank Eli Ben-Sasson and Simon Litsyn for answering our questions on coding theory.

References

[1] E. Ben-Sasson and S. Kopparty. Affine dispersers from subspace polynomials. In *STOC*, pages 65–74, 2009.

[2] J. Bourgain. On the construction of affine extractors. *Geometric And Functional Analysis*, 17(1):33–57, 2007.

[3] B. Chor, O. Goldreich, J. Håstad, J. Friedman, S. Rudich, and R. Smolensky. The bit extraction problem of t-resilient functions. In *26th Annual Symposium on Foundations of Computer Science*, pages 396–407, 1985.

[4] A. Cohen and A. Wigderson. Dispersers, deterministic amplification, and weak random sources. In *30th Annual Symposium on Foundations of Computer Science*, pages 14–19, 1989.

[5] A. Gabizon, R. Raz, and R. Shaltiel. Deterministic extractors for bit-fixing sources by obtaining an independent seed. *SICOMP: SIAM Journal on Computing*, 36(4):1072–1094, 2006.

[6] A. Gabizon and R. Shaltiel. Increasing the output length of zero-error dispersers. *Random Struct. Algorithms*, 40(1):74–104, 2012.

[7] O. Goldreich. A sample of samplers: A computational perspective on sampling. In O. Goldreich, editor, *Studies in Complexity and Cryptography*, volume 6650 of *Lecture Notes in Computer Science*, pages 302–332. Springer, 2011.

[8] C. Heegard. Partitioned linear block codes for computer memory with 'stuck-at' defects. *IEEE Transactions on Information Theory*, 29(6):831–842, 1983.

[9] J. Kamp and D. Zuckerman. Deterministic extractors for bit-fixing sources and exposure-resilient cryptography. *SIAM J. Comput.*, pages 1231–1247, 2007.

[10] A. V. Kuznetsov, T. Kasami, and S. Yamamura. An error correcting scheme for defective memory. *IEEE Trans. Inform. Theory*, 24(6):712–718, 1978.

[11] A. V. Kuznetsov and B. S. Tsybakov. Coding in a memory with defective cells. *Probl. Peredachi Inf.*, 10:52–60, 1974.

[12] L. A. Lastras-Montaño, A. Jagmohan, and M. Franceschini. Algorithms for memories with stuck cells. In *ISIT*, pages 968–972, 2010.

[13] X. Li. A new approach to affine extractors and dispersers. In *IEEE Conference on Computational Complexity*, pages 137–147, 2011.

[14] N. Nisan and D. Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, pages 43–52, 1996.

[15] A. Rao. Extractors for low-weight affine sources. In *IEEE Conference on Computational Complexity*, pages 95–101, 2009.

[16] R. Raz, O. Reingold, and S. P. Vadhan. Extracting all the randomness and reducing the error in trevisan's extractors. *J. Comput. Syst. Sci.*, 65(1):97–128, 2002.

[17] R. L. Rivest and A. Shamir. How to reuse a "write-once" memory. *Information and Control*, pages 1–19, 1982.

[18] R. Shaltiel. Recent developments in explicit constructions of extractors. *Bulletin of the EATCS*, pages 67–95, 2002.

[19] R. Shaltiel. How to get more mileage from randomness extractors. *Random Struct. Algorithms*, pages 157–186, 2008.

[20] R. Shaltiel. Dispersers for affine sources with sub-polynomial entropy. In *FOCS*, pages 247–256, 2011.

[21] R. Shaltiel. An introduction to randomness extractors. In *ICALP (2)*, pages 21–41, 2011.

[22] Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *J. ACM*, 52(2):172–216, 2005.

[23] A. Shpilka. Capacity achieving two-write wom codes. In *LATIN*, pages 631–642, 2012.

[24] L. Trevisan. Extractors and pseudorandom generators. *J. ACM*, 48(4):860–879, 2001.

[25] B. S. Tsybakov. Additive group codes for defect correction. *Prob. Peredachi Inf.*, 11:1:111–113, 1975.

[26] B. S. Tsybakov. Defect and error correction. *Prob. Peredachi Inf.*, 11:3:21–30, 1975.

[27] S. P. Vadhan. The unified theory of pseudorandomness: guest column. *SIGACT News*, pages 39–54, 2007.

[28] A. Yehudayoff. Affine extractors over prime fields. *Combinatorica*, 31(2):245–256, 2011.