Detecting and Correcting Computationally Bounded Errors: A Simple Construction Under Minimal Assumptions

Jad Silbak [*] Daniel Wichs [†]

September 18, 2024

Abstract

We study error detection and error correction in a computationally bounded world, where errors are introduced by an arbitrary polynomial time adversarial channel. We consider codes where the encoding procedure uses random coins and define two distinct variants: (1) in *randomized codes*, fresh randomness is chosen during each encoding operation and is unknown a priori, while (2) in *self-seeded codes*, the randomness of the encoding procedure is fixed once upfront and is known to the adversary. In both cases, the randomness need not be known to the decoding procedure, and there is no trusted common setup between the encoder and decoder. The encoding and decoding algorithms are efficient and run in some fixed polynomial time, independent of the run time of the adversary.

The parameters of standard codes for worst-case (inefficient) errors are limited by the Singleton bound: for rate R it is not possible to detect more than a $1 - R$ fraction of errors, or uniquely correct more than a $(1 - R)/2$ fraction of errors, and efficient codes matching this bound exist for sufficiently large alphabets. In the computationally bounded setting, we show that going beyond the Singleton bound implies *one-way functions* in the case of randomized codes and *collision-resistant hash functions* in the case of self-seeded codes. We construct randomized and self-seeded codes under these respective minimal assumptions with essentially optimal parameters over a constant-sized alphabet:

- Detection: the codes have a rate $R \approx 1$ while detecting a $\rho \approx 1$ fraction of errors.
- Correction: for any $\rho < 1/2$, the codes uniquely correct a ρ fraction of errors with rate $R \approx 1 - \rho$.

Codes for computationally bounded errors were studied in several prior works starting with Lipton (STACS '94), but all such works either: (a) need some trusted common setup (e.g., public-key infrastructure, common reference string) between the encoder and decoder, or (b) only handle channels whose complexity is a prior bounded below that of the code.

[*]Northeastern University, Email: jadsilbak@gmail.com. Research supported by the Khoury College Distinguished Postdoctoral Fellowship.

[†]Northeastern University and NTT Research. Research supported by NSF CNS-2349972 and CNS-2055510.

Contents

1 Introduction **1**
- 1.1 Our Results . . . 2
- 1.2 Our Techniques . . . 5

2 Preliminaries and Ingredients **8**
- 2.1 Notations . . . 8
- 2.2 One-Way Functions . . . 8
- 2.3 Universal One-Way Hash Functions . . . 9
- 2.4 Collision Resistance Hashing . . . 9
- 2.5 Standard Correcting/Detecting Codes . . . 10

3 Definitions for Randomized Codes and Self-Seeded Codes **11**
- 3.1 Randomized Codes . . . 11
- 3.2 Self-Seeded Codes . . . 12

4 Optimal Randomized Codes and Self-Seeded Codes **13**
- 4.1 Construction . . . 13
- 4.2 Proof of Theorem 4.1. . . . 15
  - 4.2.1 Proof of Lemma 4.3. . . . 16

5 Randomized Codes Imply One Way Functions. **17**

6 Self-Seeded Codes Imply Collision Resistance Hash Functions. **20**

1 Introduction

The theory of error detection and error correction goes back to the seminal works of Shannon [Sha48] and Hamming [Ham50]. A sender *encodes* a message and sends the resulting codeword over a noisy channel that may introduce errors by modifying a subset of the codeword symbols. A receiver gets the modified codeword and attempts to *decode* the original message. The goal of *error-detection* is to recover the original message when there are no errors or detect the presence of errors. The goal of *error-correction* is to always uniquely recover the original message. The main parameters of interest are the *rate* R, defined as the ratio of message length to codeword length, and the *relative error tolerance* ρ, defined as the ratio between the number of errors that can be detected/corrected and the codeword length.

Starting with the work of Hamming [Ham50], a large body of research studies codes for worst-case computationally unbounded (adversarial) channels that can introduce arbitrary errors patterns. The error tolerance in this setting is fully determined by the *distance* of the code, which is subject to several well-established fundamental bounds. For example, the *Singleton bound* implies that for any rate $R \geq 0$ the relative error tolerance is $\rho \leq 1-R$ for error detection and $\rho \leq (1-R)/2$ for error correction. Furthermore, there are efficient codes matching the Singleton bound for sufficiently large alphabet sizes.

Computationally Bounded Channels. We consider a setting similar to that of Hamming [Ham50], where errors are introduced by a fully adversarial channel, but with the additional natural restriction that the channel is *computationally bounded*, and can run in arbitrary polynomial (or even sub-exponential) time. This restriction is well motivated – if an error pattern takes longer than the lifetime of the universe to compute, then we can safely assume it will not occur. Our goal is to construct efficient codes that can detect/correct errors in this setting with rate vs error-tolerance trade-offs that go beyond the Singleton bound. Many prior works starting with [Lip94] and including [OPS07, MPSW10, HOSW11, GS16, SS21a, BGGZ21, SS21b, SS22, SS24, GHY20] have previously considered computationally bounded channels, either to get better parameters, or to achieve other desirable properties such as improved local/list decoding. However, all such prior works suffer from at least one of the following deficiencies:

(a) *Trusted setup:* Many works require a trusted coordinated setup between the encoding and decoding algorithms. For example, [Lip94] assumes that they share a secret key, which is unknown to the adversarial channel. Alternatively, [MPSW10] require a public-key infrastructure where the encoding algorithm has its own secret key, while the decoding algorithm (as well as the adversarial channel) gets the corresponding public-key. In both cases, the encoding algorithm is also stateful and updates its state between operations. The use of secret keys and state are significant limitations that make such codes inapplicable in many natural settings. The work of Grossman, Holmgren and Yogev [GHY20] introduces the notion of *seeded codes* that only requires a light trusted setup consisting of a *common (uniformly) random seed* shared by the encoder and decoder. They construct such seeded codes correcting $\rho \approx 1 - R$ errors over a large alphabet based on strong and non-standard cryptographic hardness assumptions – namely a certain form of *two-input correlation-intractable hash functions*, which are currently only known to exist in the random-oracle model, but have no known provably secure instantiations under standard cryptographic hardness assumptions.

(b) *Codes have higher complexity than adversary:* Many works [GS16, SS21a, SS21b, SS22, SS24] restrict the complexity (time/size or space) of the adversarial channel to be bounded by some arbitrary polynomial chosen a priori, and the complexity of the encoding/decoding algorithms then scales accordingly to some sufficiently larger polynomial. In particular, the adversarial channel is not powerful enough to evaluate the encoding/decoding algorithms in these schemes. This is a significant limitation,

since we cannot argue that such schemes protect against all errors that can arise in a computationally bounded world – a world that can compute the encoding/decoding algorithms of the scheme can also find error patterns to defeat them! In contrast, we would like to have efficient codes with some fixed polynomial run-time that protect against arbitrary polynomial (or even sub-exponential time) adversarial channels.

The different prior works and their properties are summarized in Table 1.

work	**Setup/Type**	**Channel**	**Decoding**	**Assumption**
[Lip94]	symmetric-key stateful encoding	P/poly	unique	one-way functions
[MPSW10]	public key stateful encoding	P/poly	unique	one-way functions
[GS16]	seeded + randomized	SIZE(n^c)	list	none
[SS21a]	randomized	SIZE(n^c)	list	derandomization assumption
[SS21b]	randomized	SPACE(n^δ)	unique	none
[SS22]	seeded + randomized	SIZE(n^c)	unique	none
[SS24]	randomized	SIZE(n^c)	unique	derandomization assumption
[GHY20]	seeded	P/poly	unique	correlation-intractable hash (random oracle)
This work	randomized	P/poly	unique	one-way functions
This work	self-seeded	P/poly	unique	collision-resistant hash

Table 1: ***Summary of related work.*** *Seeded codes are also known as Monte-Carlo codes in the literature and require a common random seed shared by the encoder and decoder as a trusted setup. Randomized codes require fresh on-the-fly randomness at each encoding operation. Seeded + randomized codes require both simultaneously. Codes for channels in SIZE(n^c) have correspondingly larger run-time* $\mathrm{poly}(n^c)$*, while codes for channels in P/poly have some a priori fixed polynomial run-time independent of the channel. Some works focus on the binary alphabet while others focus on larger alphabets.*

1.1 Our Results

In this work we construct error detection and error correction codes for computationally bounded channels, while simultaneously achieving: (1) no trusted setup, (2) a fixed polynomial-time code that protects against arbitrary polynomial (or even sub-exponential) time adversarial channels, (3) provable guarnatees under basic/minimal computational hardness assumptions, (4) essentially optimal trade-offs between rate and error tolerance beyond the Singleton bound for sufficiently large constant-sized alphabets.

Randomized/Self-Seeded Codes. Standard deterministic codes cannot achieve better parameters against (non-uniform) computationally bounded channels beyond what they achieve for computationally unbounded channels, and in particular, they cannot go beyond the Singleton bound. This is because a worst-case message and error pattern for which detection/correction fails can simply be hard-coded as non-uniform advice to an efficient adversary. To overcome this limitation, we therefore consider codes $(\mathrm{Enc}, \mathrm{Dec})$, where the encoding procedure Enc takes additional randomness as input. We consider two variants. In the basic variant of *randomized codes*, fresh randomness r for the encoding algorithm is chosen each time on the fly and is unknown to the adversary ahead of time. The adversary chooses a worst-case message m, gets

the resulting randomized codeword $c \leftarrow \mathrm{Enc}(m; r)$ for a fresh random r, and then chooses the error e as an arbitrary efficiently computable function of the codeword. We also consider a stronger variant called *self-seeded codes*, where the randomness r of the encoding algorithm is fixed once and for all and known to the adversary a priori. That is, the adversary gets the randomness r and then uses it to adaptively choose the message m (which fully defines the codeword $c = \mathrm{Enc}(m; r)$) along with the error e. In both cases, the fractional Hamming weight of e is bounded by the error tolerance ρ. Error detection ensures that with overwhelming probability the resulting erroneous codeword $c' = c + e$ decodes to m when there are no errors, and decodes to either m or $\perp$ when there are errors. Error correction ensures that with overwhelming probability c' decodes to m.

The main technical difference between randomized and self-seeded codes is adaptivity: in the former the message m to be encoded is chosen by the adversary before the randomness r is chosen, while in the latter the message m can adaptively depend on r.[1] Self-seeded codes allow us to fix the randomness for the code once and for all as a public seed, and afterwards the code is fully deterministic. Moreover, the seed can be chosen by the encoder unilaterally without any coordination with the decoder. For example, different software manufacturers can individually choose their own seeds to be used by the encoding algorithms in their software, while allowing for interoperability with the decoders implemented by other manufacturers without any coordination. Self-seeded codes should be contrasted to the notion of seeded codes in [GHY20], which required a shared random seed as a common trusted setup between the encoder and decoder. Self-seeded codes are the strongest notion and directly imply the seemingly incomparable notions of randomized codes and seeded codes.

Minimal Assumptions. We show that any randomized code that goes beyond the Singleton bound with error detection tolerance $\rho > 1 - R$ or error correction tolerance $\rho > (1 - R)/2$ implies *one-way functions*, while any such seeded (and therefore also self-seeded) code implies *collision-resistant hash functions*.

Main Theorem. We construct randomized and self-seeded codes going beyond the Singleton bound, with essentially optimal parameters for both error detection and error correction, under the above respective minimal assumptions.

Theorem 1.1 (Informal)**.** *Assuming one-way functions (resp. collision-resistant hash functions) there exist efficiently computable randomized (resp. self-seeded) error-detection and error-correction codes for arbitrary polynomial-time adversarial errors with the following parameters:*

- *Error Detection: For any constant $0 < \varepsilon < 1$, there is a code over a constant-sized alphabet with rate $R = 1 - \varepsilon$ that detects a $\rho = 1 - \varepsilon$ fraction of errors.*
- *Error Correction: For any constants $0 < \varepsilon < 1$ and $0 \leq \rho < 1/2$, there is a code over a constant-sized alphabet with rate $R = 1 - \rho - \varepsilon$ that uniquely corrects from a ρ fraction of errors.*

If we assume sub-exponentially secure one-way functions (resp. collision-resistant hash functions), then the above properties hold even for sub-exponential time adversarial errors.

Recall that, in the case of error-detection, the Singleton bound says that the relative error tolerance $\rho \leq 1 - R$ necessarily degrades as the rate improves when considering worst-case errors; in contrast, for computationally bounded errors, our result simultaneously achieves the best possible error tolerance $\rho \approx 1$ and rate $R \approx 1$. In the case of error correction, the Singleton bound says the maximal error tolerance is

[1] Randomized codes may also be secret-coin, where the adversary never sees the full randomness r. However, our construction will be public-coin and the randomness r can be easily recovered from the codeword c.

$\rho \leq (1-R)/2$ for worst-case errors, while for computationally bounded errors our result achieves twice as large error tolerance $\rho \approx 1-R$, subject to the upper limit $\rho < 1/2$.

Our achieved parameters are essentially optimal, up to the arbitrarily small constant ε. This is clear for error detection, where we get the best possible rate $R \approx 1$ and error tolerance $\rho \approx 1$ simultaneously. For error-correction, one cannot tolerate $\rho \geq 1/2$ fraction of errors since an efficient adversarial channel can modify 1/2 the codeword positions to a fresh encoding of a different message, causing decoding to fail with probability at least $1/2$.[2] Moreover, one cannot have rate $R > 1-\rho$ since zeroing out the first ρ fraction of positions of the codeword would remove information about the message, causing decoding to fail.

We remark that, not only are one-way functions and collision-resistant hash functions the minimal assumptions needed to achieve the above results, they are considered some of the most standard and well-studied cryptographic assumptions/primitives. In particular, one-way functions are a minimal assumption necessary for essentially any non-trivial task in cryptography and are implied by all concrete cryptographic hardness assumptions. Collision-resistant hash functions are not know to be implied by one-way function, but can be constructed from essentially every concrete cryptographic hardness assumption, such as hardness of factoring, discrete logarithms, learning with errors, etc.

The closest related prior work of [GHY20] achieves essentially the same parameters as our work, but only for the weaker notion of seeded codes, which require trusted setup, and only by assuming a strong form of "two-input correlation-intractable hash functions", which we do not know how to instantiate under any standard cryptographic hardness assumption. Our result for self-seeded codes yields a stronger primitive under significantly weaker minimal assumptions. Comparing the notions of randomized, seeded and self-seeded codes, it is clear that self-seeded codes are the strongest notion easily implying the other two, while randomized and seeded codes are seemingly incomparable. Interestingly, since we show that any non-trivial seeded codes imply collision-resistant hashing, which in turn implies self-seeded codes, we show that seeded and self-seeded codes are equivalent in terms of the needed assumptions.

Simultaneous Correction and Detection. While we stated separate results for error-detection and error-correction above, we can also simultaneously achieve a non-trivial combination of correction and detection. In particular, we can augment the previous theorem statement with the following:

> *For any constants $0 < \varepsilon < 1$ and $0 \leq \rho_c < 1/2$, there is a code over a constant-sized alphabet with rate $R = 1 - \rho_c - \varepsilon$ that simultaneously uniquely corrects from a ρ_c fraction of errors and detects a $\rho_d = 1 - \rho_c - \varepsilon$ fraction of errors.*

The above implies both of the previously stated results for error detection (by setting $\rho_c = 0$) and error correction (by ignoring ρ_d). However simultaneous error correction and detection is an interesting property, which is more than the sum of its parts. It gives a single decoding procedure that either outputs a message or $\perp$ (error detection symbol) and is guaranteed to uniquely recover the correct message if the error rate is $\leq \rho_c$ and to never output the wrong message as long as the error rate is $\leq \rho_d$. We can envision scenarios where correcting a small fraction of errors (e.g., $\rho_c = .1$) may suffice. Our result show that for such instances, we can have a code with a correspondingly large rate (e.g., $R \approx .9$) while simultaneously being able to detect a much larger fraction of errors (e.g., $\rho_d \approx .9$) at no additional cost.

Alphabet Size. We note that our results hold for large constant-size alphabets, but do not extend to the case of a binary alphabet. Giving optimal results for binary codes that detect/correct computationally bounded errors under minimal assumptions remains an interesting open problem. The results of [GHY20] for seeded

[2] The works of [Lip94, MPSW10] overcome this limitation by having a stateful and secret keyed encoding procedure to ensure that an adversarial channel cannot generate/reuse valid codewords.

error correcting codes using a strong from of correlation intractable hash functions extend to the binary case and show that one can uniquely decode computationally bounded errors with essentially the same parameters as list-decoding for worst-case errors. It remains an open problem to achieve such results for randomized or self-seeded codes under any assumptions, and/or to to get provably secure seeded codes under standard cryptographic hardness assumptions.

1.2 Our Techniques

Our techniques are quite simple in retrospect. We start by showing how to construct self-seeded codes assuming collision-resistant hashing, and then discuss how to modify these results to get analogous ones for randomized codes from one-way functions. Throughout the introduction, we use "$\approx$" to denote that two values differ by an arbitrarily small constant.

Self-Seeded Error Detection with Sub-Exponential Alphabet. We start with a naive construction of self-seeded error-detection codes that requires a sub-exponentially large alphabet size, and then show how to reduce the alphabet size to some (large) constant. Let $\mathcal{H} = \{h_r : \{0,1\}^k \to \{0,1\}^\lambda\}_{r\in\{0,1\}^\lambda}$ be a collision-resistant hash function family with output length λ (security parameter), which we can think of as $\lambda = k^\alpha$ for some small constant $\alpha > 0$. Given a random public seed $r \leftarrow \{0,1\}^\lambda$, no polynomial-time adversary can find two messages $m \neq m'$ such that $h_r(m) = h_r(m')$.

The encoding algorithm $\mathrm{Enc}(m; r)$ takes a message $m \in \{0,1\}^k$ and parses it as $n = k/w$ symbols $m = (m_1, \ldots, m_n)$ over an alphabet $\Sigma_{\mathrm{data}} = \{0,1\}^w$ for some large $w = O(\lambda)$. It then "folds in" the seed r and hash of the message $h_r(m)$ into each message symbol to yield the codeword

$$c = (\ [m_1, r, h_r(m)]\ ,\ [m_2, r, h_r(m)]\ ,\ \ldots\ ,\ [m_n, r, h_r(m)]\ ) \in \Sigma^n$$

interpreted as n symbols over the larger alphabet $\Sigma = \{0,1\}^{w+2\cdot\lambda}$. The decoding algorithm $\mathrm{Dec}(c')$ gets

$$c' = (\ [m'_1, r'_1, y'_1]\ ,\ [m'_2, r'_2, y'_2]\ ,\ \ldots\ ,\ [m'_n, r'_n, y'_n]\ ) \in \Sigma^n,$$

which gives a candidate message $m' = (m'_1, \ldots, m'_n)$. It checks that the second part of each codeword symbol has the same common seed value $r'_1 = \cdots = r'_n$ and that the third part of each symbol has the same common hash value $y'_i = h_{r'_i}(m')$; if so it outputs m' else $\perp$. The code achieves rate $R = w/(w + 2\lambda) \approx 1$ by choosing a sufficiently large $w = O(\lambda)$. Moreover, it achieves error-tolerance $\rho = 1 - 1/n$. Unless the adversary changes *every* symbol of the codeword c, or finds a collision, decoding is guaranteed to output the correct message m or $\perp$. To see this, assume at least one symbol of the codeword remains unchanged and decoding outputs $m' \neq m$ for some $m' \neq \perp$. Then it must be the case that $r'_i = r, y'_i = h_r(m') = h_r(m)$ for all i and therefore the this gives a collision $m \neq m'$ such that $h_r(m) = h_r(m')$.

Self-Seeded Error Detection with Constant Alphabet. We reduce the alphabet size to a constant by applying a standard error-correcting code $(\mathrm{Enc}_{\mathrm{ctrl}}, \mathrm{Dec}_{\mathrm{ctrl}})$ over a constant sized alphabet to the small "control" value $(r, h_r(m))$. In particular, for some constants $w > v$, we use a code with alphabet $\Sigma_{\mathrm{ctrl}} = \{0,1\}^v$ such that $\mathrm{Enc}_{\mathrm{ctrl}} : \{0,1\}^{2\lambda} \to \Sigma^n_{\mathrm{ctrl}}$ maps a 2λ-bit message to a codeword of length $n = k/w$. The corresponding rate of the code is very small $R_{\mathrm{ctrl}} = O(\lambda/k) = o(1)$, but we will want large distance $\delta_{\mathrm{ctrl}} \approx 1$. Such codes can be constructed by concatenating Reed-Solomon codes with a brute-force inner code over the alphabet Σ_{ctrl} (see Theorem 2.12). We do not need efficient error-decoding for now, but only require an efficient way to recognize and valid codewords and invert them without errors. The encoding algorithm of our self-seeded code $\mathrm{Enc}(m; r)$ first computes $c_{\mathrm{ctrl}} = (c_{\mathrm{ctrl},1}, \ldots, c_{\mathrm{ctrl},n}) = \mathrm{Enc}_{\mathrm{ctrl}}((r, h_r(m))) \in \Sigma^n_{\mathrm{ctrl}}$.

It interprets $m \in \{0,1\}^k$ as $m = (m_1, \ldots, m_n) \in \Sigma_{\text{data}}^n$ consisting of $n = k/w$ symbols over the alphabet $\Sigma_{\text{data}} = \{0,1\}^w$ and then "folds" the two vectors together to yield the final codeword

$$c = (\ [m_1, c_{\text{ctrl},1}]\ ,\ \ldots\ ,\ [m_n, c_{\text{ctrl},n}]\ ) \in \Sigma^n,$$

interpreted as n symbols in over the alphabet $\Sigma = \Sigma_{\text{data}} \times \Sigma_{\text{ctrl}} = \{0,1\}^{v+w}$. The decoding algorithm $\text{Dec}(c')$ gets

$$c' = (\ [m'_1, c'_{\text{ctrl},1}]\ ,\ \ldots\ ,\ [m'_n, c'_{\text{ctrl},n}]\ ) \in \Sigma^n,$$

and recovers a candidate message $m' = (m'_1, \ldots, m'_n)$ as well as a candidate $c'_{\text{ctrl}} = (c'_{\text{ctrl},1}, \ldots, c'_{\text{ctrl},n})$. It checks that c'_{ctrl} is a valid codeword and if so recovers (r', y') such that $c'_{\text{ctrl}} = \text{Enc}_{\text{ctrl}}(r', y')$. If c'_{ctrl} is not a valid codeword or $h_{r'}(m') \neq y'$, it outputs $\perp$ else it outputs m'. Note that this generalizes the previous naive idea with a sub-exponentially large alphabet, which corresponds to using a repetition code for the control code. The rate of the overall code is $R = w/(w+v)$ which can be made arbitrarily close to 1 by choosing a sufficiently large constant $w > v$. The error-tolerance is $\rho = \delta_{\text{ctrl}} \approx 1$. In particular, if c' is within distance ρ of c and c'_{ctrl} is a valid codeword then it must be the case that $c'_{\text{ctrl}} = c_{\text{ctrl}}$ and hence $r' = r, y' = h_r(m)$. In this case, the only way that that decoding can output some $m' \neq m$ is if the values form a hash collision $h_r(m') = h_r(m)$.

Self-Seeded Error Correction. To achieve error-correction, we use essentially the same construction as before, but additionally encode the data m using an efficiently list-decodable error-correcting code $(\text{Enc}_{\text{data}}, \text{Dec}_{\text{data}})$ over a constant size alphabet $\Sigma_{\text{data}} = \{0,1\}^w$ for some large constant w. For any constant $\rho < 1/2$ we want this code to have rate $R_{\text{data}} \approx 1 - \rho$ and to be list-decodable from a ρ fraction of errors. This can be accomplished using the codes of [GR08]. Moreover, we will now want the code $(\text{Enc}_{\text{ctrl}}, \text{Dec}_{\text{ctrl}})$ to efficiently uniquely decode from ρ errors, which can be achieved with the same parameters as previously. Our self-seeded code $\text{Enc}(m; r)$ computes $c_{\text{ctrl}} = (c_{\text{ctrl},1}, \ldots, c_{\text{ctrl},n}) = \text{Enc}_{\text{ctrl}}((r, h_r(m))) \in \Sigma_{\text{ctrl}}^n$ as before, and also $c_{\text{data}} = (c_{\text{data},1}, \ldots, c_{\text{data},n}) = \text{Enc}_{\text{data}}(m) \in \Sigma_{\text{data}}^n$. It then "folds" the two codewords together to yield the final codeword

$$c = (\ [c_{\text{data},1}, c_{\text{ctrl},1}]\ ,\ \ldots\ ,\ [c_{\text{data},n}, c_{\text{ctrl},n}]\ ) \in \Sigma^n$$

where $\Sigma = \Sigma_{\text{data}} \times \Sigma_{\text{ctrl}} = \{0,1\}^{v+w}$. The decoding algorithm $\text{Dec}(c')$ gets

$$c' = (\ [c'_{\text{data},1}, c'_{\text{ctrl},1}]\ ,\ \ldots\ ,\ [c'_{\text{data},n}, c'_{\text{ctrl},n}]\ ) \in \Sigma^n.$$

It applies list-decoding on the data part c'_{data} of the received value c to recover a polynomial list of candidate messages $\{m^{(i)}\} = \text{Dec}_{\text{data}}(c'_{\text{data}})$. It also uniquely decodes $(r', y') = \text{Dec}_{\text{ctrl}}(c'_{\text{ctrl}})$. Then it outputs the first value $m^{(i)}$ such that $h_{r'}(m^{(i)}) = y'$, or outputs $\perp$ if none is found or if either of the decoding procedures fail. The rate of the overall code is $R = R_{\text{data}} \cdot (w+v)/w \approx 1 - \rho$ by choosing sufficiently large constants $w > v$. To analyze the error-correction, first observe that if the received value c' within relative distance ρ of $c = \text{Enc}(m; r)$ then, by the correctness of list-decoding of the "data code", the correct message m will be in the decoded list with $m = m^{(i)}$ for some i, and by the correctness of unique decoding of the "control code" we have $(r', y') = (r, h_r(m))$. So the only way that error correction can fail is if there is another $m^{(j)} \neq m$ in the list such that $h_r(m^{(j)}) = h_r(m)$, but this gives a collision on the hash function.

Randomized Codes from One-Way Functions. Our constructions of *randomized* codes with error-detection and error-correction are identical to the above constructions of *self-seeded* codes, but instead of requiring the hash family h_r to be collision-resistant, we only need it to be a *universal one-way hash function (UOWHF)*.

A UOWHF ensures that a polynomial time adversary that selectively chooses a message m first and then gets the seed r cannot find a collision $m' \neq m$ such that $h_r(m) = h_r(m')$ except with negligible probability. Such UOWHFs can be constructed from one-way functions [NY89, Rom90]. Recall that the only difference between self-seeded and randomized codes is whether the encoded message m can be chosen adaptivley depending on the randomness r or selectively before r is chosen, which exactly matches the difference between collision-resistant hash functions and UOWHFs.

Necessity of Assumptions. We show that self-seeded codes beating the Singleton bound, with error-detection tolerance $\rho > 1 - R$ or error-correction tolerance $\rho > (1 - R)/2$, imply collision-resistant hash functions (CRHFs). Start by considering a self-seeded error-detection code $(\mathrm{Enc}, \mathrm{Dec})$ with error-detection tolerance ρ. We claim that for every fixed subset S of $1 - \rho$ fraction of codeword positions, the function $h_r(m) = \mathrm{Enc}(m; r)[S]$ that outputs the codeword symbols of $\mathrm{Enc}(m; r)$ in the positions indexed by S is collision-resistant. In particular, if there is an efficient adversary that, given r, can find a collision $m \neq m'$ such that $h_r(m) = h_r(m')$, it means that $\mathrm{Enc}(m; r)$ and $\mathrm{Enc}(m'; r)$ are at distance $< \rho$ and therefore such an adversary can break error detection by choosing the message m and modifying $\mathrm{Enc}(m; r)$ to $\mathrm{Enc}(m'; r)$. Moreover, if the rate of the code exceeds the Singleton bound with $R > 1 - \rho$, then the functions h_r are compressing, making them CHRFs. To get the analogous result for error-correction, we simply note that ρ-error-correction implies (2ρ)-error-detection. In particular, if an efficient adversary given r can find two messages $m_0 \neq m_1$ such that $\mathrm{Enc}(m_0; r)$ and $\mathrm{Enc}(m_1; r)$ are within relative distance 2ρ then it can find a "midpoint" c' which is at relative distance ρ from both $\mathrm{Enc}(m_0; r)$ and $\mathrm{Enc}(m_1; r)$. For one of $b \in \{0, 1\}$, modifying $\mathrm{Enc}(m_b; r)$ to c' necessarily causes decoding to output the incorrect message m_{1-b}.

We also show that randomized error-detection and error-correction codes beating the Singleton bound imply one-way functions. The argument is somewhat different from above since we don't want to assume that $\mathrm{Enc}(m; r)$ reveals r in the full. Instead, we rely on the fact that, if one-way functions don't exist, then we can efficiently sample "almost uniform" inverses of any efficient function [IL89]. Consider a random codeword $c = \mathrm{Enc}(m; r)$ for a random message m and randomness r. For any subset S of $1 - \rho$ fraction of codeword positions we can efficiently find an almost uniform inverse (m', r') of the function $f(m, r) = \mathrm{Enc}(m; r)[S]$ that outputs the codeword symbols in positions S. This means that $c = \mathrm{Enc}(m; r)$ and $c' = \mathrm{Enc}(m'; r')$ agree on the positions in S and therefore have relative distance $\leq \rho$. If the rate of the code exceeds the Singleton bound with $R > 1 - \rho$, then there is a non-negligible probability that $m \neq m'$ since f loses information about m. Therefore the channel that changes c to c' is efficient and defeats error detection by introducing ρ-fraction errors. Similarly the channel that changes a random subset of $\rho/2$ positions outside of S from c to c' defeats error correction since decoding is almost as likely to output m' as m.

Organization of the Paper.

In Section 2 we give general notations and state the ingredients that we will use. In Section 3 we give formal definitions for randomized codes and self-seeded codes. In Section 4 we state and prove our main theorems. In Section 5 we show that randomized codes imply one-way functions and in Section 6 we prove that self-seeded codes imply collision resistance hash functions.

Acknowledgement.

We are very grateful to Ronen Shaltiel and Tal Yankovitz for very useful discussions.

2 Preliminaries and Ingredients

2.1 Notations

All logarithms are taken in base 2. Let poly stand for the set of all polynomials. Let PPT stand for non-uniform probabilistic polynomial-time. We note that while we rely on the non-uniform model of computation by default, all our results also hold if we were to define PPT in the uniform model. We use sans-serif (e.g., A) for PPT algorithms. A function $\nu\colon \mathbb{N} \mapsto [0,1]$ is *negligible*, denoted $\nu(k) = \mathrm{neg}(k)$, if $\nu(k) < 1/p(k)$ for every $p \in \mathrm{poly}$ and large enough k. For a set S, let $x \leftarrow S$ denote that x was drawn uniformly from S. Given a distribution P, we use $x \leftarrow P$ to denote that x is chosen according to P.

Hamming distance. The Hamming distance between $x, y \in [q]^n$ is $\Delta(x,y) = |\{i : x_i \neq y_i\}|$. The relative Hamming distance between $x, y \in [q]^n$ is $\delta(x,y) = \frac{\Delta(x,y)}{n}$.

Distributions and random variables. The statistical distance (also known as, variation distance) of two distributions P and Q over a discrete domain X is defined by $\mathsf{SD}(P,Q) = \max_{S \subseteq X} |P(S) - Q(S)|$.

Definition 2.1 (Min-entropy)**.** *A random variable X has min-entropy k, denoted $H_\infty(X) \geq k$, if $\max_x \mathbf{Pr}[X = x] \leq 2^{-k}$.*

We will also use the following lemma about average min-entropy:

Definition 2.2 (Average min-entropy)**.** *Let (X,Y) be a pair of random variables. The average min-entropy of X conditioned on Y is:*

$$\tilde{H}_\infty(X|Y) = -\log\left[\mathop{\mathbf{E}}_{y \leftarrow Y}\left[2^{-H_\infty(X|Y=y)}\right]\right]$$

Average min-entropy gives a bound on the probability that an adversary that gets the value of Y will guess the value of X in a single attempt. We will use the following useful lemma that connects average min-entropy and min-entropy.

Lemma 2.3 ([DORS08, Rey11])**.** *$\tilde{H}_\infty(X|Y) \geq H_\infty(X,Y) - v$, where 2^v is the number of elements in support of Y.*

2.2 One-Way Functions

Definition 2.4 (One-way functions)**.** *A* PPT *algorithm $\mathsf{F}\colon \{0,1\}^* \to \{0,1\}^*$ is a* **one-way function** *if for every* PPT A *and every sufficiently large $k \in \mathbb{N}$,*

$$\mathop{\mathbf{Pr}}_{x \leftarrow \{0,1\}^k}\left[\mathsf{A}(1^k, \mathsf{F}(x)) \in \mathsf{F}^{-1}(\mathsf{F}(x))\right] \leq \mathrm{neg}(k).$$

Distributional one-way functions. We will also use the classical result of Impagliazzo and Luby on distributional one-way functions implying one-way functions.

Definition 2.5 (Distributional One-way functions)**.** *A* PPT *algorithm $\mathsf{F}\colon \{0,1\}^* \to \{0,1\}^*$ is a* **distributional one-way function** *if there exists a constant $c > 0$ such that for every* PPT A *and every sufficiently large $k \in \mathbb{N}$,*

$$\mathsf{SD}\Big(\big(\mathsf{A}(1^k, \mathsf{F}(X)), \mathsf{F}(X)\big), \big(X, \mathsf{F}(X)\big)\Big) \geq 1/k^c.$$

where X is sampled uniformly from $\{0,1\}^k$.

Theorem 2.6 (Distributional one-way functions imply one-way functions, [IL89])**.** *If there exists a distributional one-way function then there exists a one-way function.*

2.3 Universal One-Way Hash Functions

Definition 2.7 (Universal one-way hash functions)**.** *For a security parameter $k \in \mathbb{N}$. A family of functions $\mathcal{H} = \{h_s \colon \{0,1\}^{n(k)} \to \{0,1\}^{\ell(k)}\}_{s \in \{0,1\}^{\ell(k)}}$ is a family of* **universal one-way hash functions** *(UOWHFs) if it satisfies:*

1. *Efficiency: Given $s \in \{0,1\}^{\ell(k)}$ and $x \in \{0,1\}^{n(k)}$, $h_s(x)$ can be evaluated in time $\mathrm{poly}(n(k), \ell(k))$.*
2. *Shrinking: $\ell(k) < n(k)$.*
3. *Target Collision Resistance: For every PPT A and sufficiently large $k \in \mathbb{N}$, if we consider the following randomized experiment:*
   - $(x, \mathrm{state}) \leftarrow \mathsf{A}(1^k) \in \{0,1\}^{n(k)} \times \{0,1\}^*$,
   - *Sampling* $s \leftarrow \{0,1\}^{\ell(k)}$,
   - $x' \leftarrow \mathsf{A}(\mathrm{state}, s)$

   Then, $\mathbf{Pr}\left[x \neq x' \wedge h_s(x) = h_s(x')\right] \leq \mathrm{neg}(k)$.

The notion of universal one-way hash functions (UOWHF) was first introduced by Naor and Yung [NY89]. Rompel [Rom90] gave the first UOWHF construction from arbitrary one-way functions.

Theorem 2.8 (One-way functions imply universal one-way hash function, [NY89, Rom90])**.** *There is a black-box construction of universal one-way hash functions (UOWHFs) from one-way functions (OWFs). In particular, assuming OWFs, for any arbitrarily large constant c and arbitrarily small constant $\delta > 0$ there is a UOWHF with input length $n(k) = k^c$ and output/seed length $\ell(k) = k^\delta$.*

2.4 Collision Resistance Hashing

Definition 2.9 (Collision resistance hash functions (CRHF))**.** *For a security parameter $k \in \mathbb{N}$. A family of functions $\mathcal{H} = \{h_s \colon \{0,1\}^{n(k)} \to \{0,1\}^{\ell(k)}\}_{s \in \{0,1\}^{\ell(k)}}$ is a family of* **collision resistance hash functions (CRHFs)** *if it satisfies:*

1. *Efficiency: Given $s \in \{0,1\}^{\ell(k)}$ and $x \in \{0,1\}^{n(k)}$, $h_s(x)$ can be evaluated in time $\mathrm{poly}(n(k), \ell(k))$.*
2. *Shrinking: $\ell(k) < n(k)$.*
3. *Collision Resistance: For every PPT A and sufficiently large $k \in \mathbb{N}$,*

$$\Pr_{\substack{s \leftarrow \{0,1\}^k, \\ x, x' \leftarrow \mathsf{A}(s)}} \left[x \neq x' \wedge h_s(x) = h_s(x')\right] \leq \mathrm{neg}(k).$$

We mention that the parameters of CRHFs above do not matter much: any "non-trivial" CRHF implies an "arbitrarily good" CRHF. That is, start with any CRHF with arbitrarily large polynomial seed/output length $\ell(k)$ that has even 1-bit compression $n(k) = \ell(k) + 1$. Then, for any arbitrarily small constant $\delta > 0$ and arbitrarily large constant c, we can also get a CRHF with small seed/output length $\ell(k) = k^\delta$ and large input length $n(k) = k^c$. Therefore, when we assume the existence of CRHFs we without loss of generality assume the above "arbitrarily good" CRHF parameters.[3]

[3] We can decrease the seed/output length by changing the security parameter from k to k^ε for a sufficiently small ε and noting that poly/negl in k^ε is the same as poly/negl in k. We can increase the input length arbitrarily using the Merkle-Damgard transform.

2.5 Standard Correcting/Detecting Codes

We now define error-correcting codes for information-theoretic adversaries also called Hamming channels. In this section, a code is a pair $(\mathrm{Enc}, \mathrm{Dec})$ of deterministic encoding and decoding algorithms, and different notions are obtained by considering the requirements of the decoding algorithm.

Definition 2.10 (Standard codes for Hamming channels)**.** *Let k, n, q be parameters and let $\mathrm{Enc} : \{0,1\}^k \to [q]^n$ be a function. We say that Enc is an encoding function for a code that:*

- **decodes d errors**, *if there exists a function $\mathrm{Dec} : [q]^n \to \{0,1\}^k$ such that for every $m \in \{0,1\}^k$ and every $v \in [q]^n$ with $\Delta(\mathrm{Enc}(m), v) \leq d$, we have $\mathrm{Dec}(v) = m$.*
- **L-list-decodes d errors**, *if the function Dec is allowed to output a list of size at most L, and for every $m \in \{0,1\}^k$ and every $v \in [q]^n$ with $\Delta(\mathrm{Enc}(m), v) \leq d$, we have $m \in \mathrm{Dec}(v)$.*
- **detects d errors**, *if there exists a function $\mathrm{Dec} : [q]^n \to \{0,1\}^k \cup \{\perp\}$ such that for every $m \in \{0,1\}^k$, $\mathrm{Dec}(\mathrm{Enc}(m)) = m$ and for every $v \in [q]^n$ such that $v \neq \mathrm{Enc}(m)$ and $\Delta(\mathrm{Enc}(m), v) \leq d$, we have $\mathrm{Dec}(v) = \perp$.*

The rate of the code is the ratio of the message length and output length of Enc, where both lengths are measured in bits. That is the rate $R = \frac{k}{n \cdot \log q}$.

A code $(\mathrm{Enc}, \mathrm{Dec})$, is **explicit** *if Enc and Dec run in polynomial time. (Naturally, this makes sense only for a family of encoding and decoding/detecting functions with varying message length k block length $n(k)$, and alphabet sizes $q(k)$).*[4]

We will also use the following construction by Guruswami and Rudra [GR08] for optimal list-decodable codes over a constant alphabet.

Theorem 2.11 ([GR08])**.** *For every $0 < R < 1$, every sufficiently small $\varepsilon > 0$, there is an explicit family of codes over an alphabet of size $2^{O(\varepsilon^{-4} \log(\frac{1}{\varepsilon}))}$ that have rate at least R and which can be list decoded up to a fraction $1 - R - \varepsilon$ of errors in polynomial time, with a list of size $n^{O(\varepsilon^{-1} \log(\frac{1}{\varepsilon}))}$.*[5]

It is folklore that for every $\varepsilon > 0$ by using code concatenation it is possible to construct standard codes with distance $d \geq 1 - \varepsilon$ and rate $R = o(1)$ for any sufficiently large k with an alphabet size that depends on ε. We make this formal in the following Theorem (see for example the book by Guruswami, Rudra, and Sudan [GRS] for more details).

Theorem 2.12. *For every sufficiently small $0 < \varepsilon$, and every computable function $n : \mathbb{N} \mapsto \mathbb{N}$ such that for every sufficiently large k, $n(k) > k^c$, (for a universal constant $c > 1$) there exists an explicit standard code $(\mathrm{Enc}, \mathrm{Dec})$, for $\mathrm{Enc} \colon \{0,1\}^k \to [q]^{n(k)}$ and $\mathrm{Dec} \colon [q]^{n(k)} \to \{0,1\}^k$ with alphabet $q = 2^{(1/\varepsilon^3)}$ such that the code has distance $d(k) > n(k) - \varepsilon \cdot n(k)$ and (uniquely) decodes from $d(k)/2 - 1$ errors.*

[4] We point out that traditionally codes are indexed by the codeword length (that is, k and q are chosen as a function of the codeword size n). However, for the sake of consistency with the rest of our definitions, we chose to define the codeword length n and alphabet q as a function of the input size k.

[5] We remark that while Theorem 2.11 does not explicitly mention that the rate can be achieved for every sufficiently large message size k, and instead just claims that it holds for infinitely often choices of k. A careful examination of the construction and proof by [GR08] reveals that the argument and proof hold for every sufficiently large k, given a suitable expander graph. An explicit construction for such graphs were given for every sufficiently large k in [KMRZS17] (see Lemma 2.7 for a formal proof). Thus, we assume that Theorem 2.11 holds for every sufficiently large k.

3 Definitions for Randomized Codes and Self-Seeded Codes

In this section, we formally define randomized codes and self-seeded codes, and their related notions of error detection and correction.

3.1 Randomized Codes

We start by giving a formal definition of randomized encoding and decoding algorithms. This syntax will be also shared with our notion of self-seeded codes defined in the next section.

Definition 3.1 (Randomized codes)**.** *Let $n, q : \mathbb{N} \mapsto \mathbb{N}$ be poly-time computable functions. An (n, q)-***randomized** *code is a pair $(\mathrm{Enc}, \mathrm{Dec})$ such that:*

- Enc *is a* PPT *algorithm that on input $m \in \{0,1\}^k$ outputs a string in $c \in [q(k)]^{n(k)}$. We use $\mathrm{Enc}_r(m)$ to denote the instantiation of $\mathrm{Enc}(m)$ when using the string r as random coins.*
- Dec *is a* PPT *algorithm that on input $z \in [q(k)]^{n(k)}$ outputs a string $\widehat{m} \in \{0,1\}^k \cup \{\perp\}$.*

For a fixed $k \in \mathbb{N}$, the rate of the code for k is defined to be $R_k = \frac{k}{\log(q) \cdot n(k)}$. When $R = \lim_{k \to \infty} R_k \in [0,1]$ is well defined we say that R is the rate of the code. We omit all or a subset of the functions n and q when they are clear from the context.

We now define a notion of reliable codes for randomized codes, which essentially states that it is possible to recover the message given a codeword (that was not damaged by error).

Definition 3.2 (Codes with reliable decoding)**.** *A randomized code $(\mathrm{Enc}, \mathrm{Dec})$ is said to be* **perfectly reliable** *if for every $k \in \mathbb{N}$ and $m \in \{0,1\}^k$, $\mathbf{Pr}\left[\mathrm{Dec}(\mathrm{Enc}(m)) = m\right] = 1$.*

Intuitively, in randomized codes, each time the sender wants to send a message, it samples fresh randomness on the fly and uses it to encode the message. We make this formal in the definition below.

Definition 3.3 (Error detection/correction for randomized codes)**.** *Let $n, \ell, d : \mathbb{N} \mapsto \mathbb{N}$ be poly-time computable functions and let $(\mathrm{Enc}, \mathrm{Dec})$ be a randomized code. For a* PPT *algorithm A and $k \in \mathbb{N}$, consider the following randomized experiment:*

1. $(m, \mathrm{state}) \leftarrow \mathsf{A}(1^k) \in \{0,1\}^k \times \{0,1\}^*$
2. *Compute $c \leftarrow \mathrm{Enc}(m)$ (by sampling $r \leftarrow \{0,1\}^{\ell(k)}$ and computing $\mathrm{Enc}_r(m)$).*
3. $z \leftarrow \mathsf{A}(m, \mathrm{state}, c)$

We say that the code:

- *is* **reliable** *if for every* PPT *algorithm A and every sufficiently large $k \in \mathbb{N}$:* $\mathbf{Pr}\left[\mathrm{Dec}(c) \neq m\right] \leq \mathrm{neg}(k)$.[6]
- **detects** *d-errors if it is a* **reliable** *code, and for every* PPT *algorithm A and every sufficiently large $k \in \mathbb{N}$:* $\mathbf{Pr}\left[\Delta(c, z) \leq d(k) \wedge \mathrm{Dec}(z) \notin \{\perp, m\}\right] \leq \mathrm{neg}(k)$.
- **corrects** *d-errors if for every* PPT *algorithm A and every sufficiently large $k \in \mathbb{N}$:* $\mathbf{Pr}\left[\Delta(c, z) \leq d(k) \wedge \mathrm{Dec}(z) \neq m\right] \leq \mathrm{neg}(k)$.

[6] This is a weaker variant of Definition 3.2. Our construction achieves the stronger variant of perfect reliability. However, our lower bounds results are stronger since they apply to this weaker notion.

For a constant $\rho \in [0,1]$, the code detects or corrects ρ-relative errors, if for every sufficiently large $k \in \mathbb{N}$, $\rho \leq d(k)/n(k)$.

Note that by definition if a code corrects d-errors for $d > 0$ then the code is reliable. In addition, our definition of error detection permits also error correction (that is, when errors are induced, the decoding algorithm can either return $\perp$ or successfully decode and return the original message m). By doing so we can achieve simultaneous error detection and correction (that is, the same decoding function can at the same time satisfy the notion of detection and correction defined above).

3.2 Self-Seeded Codes

We now formally, define the notion of self-seeded codes. A self-seeded code is a randomized code that shares the same syntax as defined in 3.1. However, a self-seeded code provides a strictly stronger notion of security as compared to the notion of error correction and detection for randomized codes stated above in Definition 3.3.

Definition 3.4 (Error detection/correction for self-seeded codes)**.** *Let $n, \ell, d : \mathbb{N} \mapsto \mathbb{N}$ be poly-time computable functions. A randomized code is said to be a* **self-seeded** *code if the following holds: For a* PPT *algorithm* A *and $k \in \mathbb{N}$, consider the following randomized experiment:*

1. $r \leftarrow \{0,1\}^{\ell(k)}$
2. $(m, z) \leftarrow \mathsf{A}(r)$

We say that the code:

- *is* **reliable** *if for every* PPT *algorithm* A *and every sufficiently large $k \in \mathbb{N}$:* $\mathbf{Pr}\left[\mathrm{Dec}(\mathrm{Enc}_r(m)) \neq m\right] \leq \mathrm{neg}(k)$.
- **detects** *d-errors if it is a* **reliable** *code, and for every* PPT *algorithm* A *and every sufficiently large $k \in \mathbb{N}$:* $\mathbf{Pr}\left[\Delta(\mathrm{Enc}_r(m), z) \leq d(k) \wedge \mathrm{Dec}(z) \notin \{\perp, m\}\right] \leq \mathrm{neg}(k)$.
- **corrects** *d-errors if for every* PPT *algorithm* A *and every sufficiently large $k \in \mathbb{N}$:* $\mathbf{Pr}\left[\Delta(\mathrm{Enc}_r(m), z) \leq d(k) \wedge \mathrm{Dec}(z) \neq m\right] \leq \mathrm{neg}(k)$.

For a constant $\rho \in [0,1]$, the code detects or corrects ρ-relative errors, if for every sufficiently large $k \in \mathbb{N}$, $\rho \leq d(k)/n(k)$.

Self-seeded code enables the sender to sample its randomness once, fix it, and then use this same randomness for encoding multiple messages (this contrasts with the standard notion of randomized codes, which requires the sender to generate fresh randomness for each message in real time). We stress that after being fixed, this randomness is known to any potential adversary but is *not* given to the decoding algorithm.[7] Still, we require that no efficient adversary can find a message and a close error vector that leads to a decoding error.

Indeed, self-seeded codes offer a stronger adaptive notion of security, where the adversary can choose the message m adaptively based on the randomness fixed by the sender. In contrast, standard randomized codes require the sender to choose fresh randomness after the adversary has decided on the message.

[7] We note that our notion of self-seeded codes is strictly stronger than the notion of "seeded codes" defined in [GHY20]. Seeded codes (unlike self-seeded codes), requires the encoding and decoding algorithms to share a random seed. Thus, unlike self-seeded codes, seeded codes necessitate a trusted setup between the encoding and decoding.

4 Optimal Randomized Codes and Self-Seeded Codes

We now state our main theorems for randomized codes and self-seeded codes. Our results show that it is possible to construct codes that bypass the Singleton bound for computationally bounded channels under minimal cryptographic assumptions. Specifically, for the randomized codes construction, we assume the existence of one-way functions, and for the self-seeded codes construction, we assume the existence of a collision resistance hash function.

Main theorem. We state our result for randomized codes and self-seeded codes together in the following theorem.

Theorem 4.1 (Randomized codes [resp., self-seeded codes] with constant alphabet size)**.** *Assuming the existence of one-way functions [resp., collision resistance hash functions], the following holds: For every constant $0 \leq p < 1/2$ and every sufficiently small constant $0 < \varepsilon$, there exists a (q, n)-randomized [resp., (q, n)-self-seeded code] code with rate $R = 1 - p - \varepsilon$ that simultaneously (uniquely) corrects p relative errors and detects from $1 - p - \varepsilon$ relative errors. Moreover, the alphabet size q is a constant that depends on ε.*

The proof of Theorem 4.1 appears in Section 4.2 and relies on our construction described in Section 4.1.

Remark 4.2. *If the underlying cryptographic primitives assumed in our theorem statements (that is, one-way functions for randomized codes and collision resistance hash functions for self-seeded codes) have sub-exponential security, then our constructed codes in Theorem 4.1 will also have sub-exponential failure probability for correction and detection against channels that are computable by sub-exponential size circuits. This follows since all our reductions run in polynomial time and have a polynomial loss in advantage. However, for the sake of simplicity, we focus on the case of polynomial/negligible security in our formal statements and proofs.*

4.1 Construction

In this section, we describe the construction that we will use to prove Theorem 4.1.

We now describe the encoding and decoding algorithms. In Figure 1, we state the parameters and ingredients that are needed to define our encoding and decoding algorithms $(\mathrm{Enc}, \mathrm{Dec})$ that are specified in Figure 2. The encoding and decoding algorithms are essentially the same for randomized codes and for self-seeded codes. In both cases the encoding algorithm Enc takes as input a message m and some value r. For randomized codes, a fresh random r is chosen every time we send a new message, and for the self-seeded codes r is sampled once (and is given to any possible adversary), and this single fixed r could be used in the encoding of multiple messages.

In both cases, the decoding algorithm Dec only takes as input a possibly damaged codeword z. In particular, the decoding algorithm is not provided with the seed r.

Figure 1: Parameters and Ingredients for folding construction.

List of ingredients and parameters: Let $q, q_{\mathrm{data}}, q_{\mathrm{ctrl}}, d_{\mathrm{data}}, d_{\mathrm{ctrl}}, \ell, n : \mathbb{N} \mapsto \mathbb{N}$ and $R_{\mathrm{data}} : \mathbb{N} \mapsto \mathbb{R}$ be poly-time computable functions.

- **Data code:** A family of standard codes $(\mathrm{Enc}_{\mathrm{data}}, \mathrm{Dec}_{\mathrm{data}})$ such that $\mathrm{Enc}_{\mathrm{data}} \colon \{0,1\}^k \to [q_{\mathrm{data}}]^n$ list-decodes from d_{data} errors with rate R_{data} and polynomial list size.
- **A hash family:** $\mathcal{H} = \{h_s \colon \{0,1\}^k \to \{0,1\}^\ell\}_{s \in \{0,1\}^\ell}$ that is efficiently computable. Where we require that $2\ell \coloneqq \ell' \leq k^{c_\ell}$ (where $c_\ell < 1$ is a fixed universal constant).
- **Control code:** Code with large distance: A family of standard codes $(\mathrm{Enc}_{\mathrm{ctrl}}, \mathrm{Dec}_{\mathrm{ctrl}})$, such that $\mathrm{Enc}_{\mathrm{ctrl}} \colon \{0,1\}^{\ell'} \to [q_{\mathrm{ctrl}}]^n$ has distance larger than $d_{\mathrm{ctrl}} > 2 \cdot d_{\mathrm{data}} + 1$ and efficiently uniquely decodes from $d_{\mathrm{ctrl}}/2 - 1$ errors. (Note that we require the codeword size n to be the same in $\mathrm{Enc}_{\mathrm{data}}$ and $\mathrm{Enc}_{\mathrm{ctrl}}$).

Figure 2: Encoding and decoding algorithm

Encoding algorithm: We define an encoding function $\mathrm{Enc} : \{0,1\}^k \times \{0,1\}^\ell \to [q]^n$:

Input: A message $m \in \{0,1\}^k$. And coins $r \in \{0,1\}^\ell$.

Output: A codeword $c \in [q]^n$ for $q \coloneqq q_{\mathrm{ctrl}} \cdot q_{\mathrm{data}}$.

Operation :

- **Encode the data:** Compute $c_{\mathrm{data}} = \mathrm{Enc}_{\mathrm{data}}(m) \in [q_{\mathrm{data}}]^n$.
- **Hash the encoded data:** Compute $y = h_r(m)$ and set $w \coloneqq (y, r)$.
- **Encode the control:** Compute $c_{\mathrm{ctrl}} = \mathrm{Enc}_{\mathrm{ctrl}}(w) \in [q_{\mathrm{ctrl}}]^n$
- **Fold the control and data codes:** Compute $c \in ([q_{\mathrm{data}}] \times [q_{\mathrm{ctrl}}])^n$, where for every $i \in [n]$, $c_i = ((c_{\mathrm{data}})_i, (c_{\mathrm{ctrl}})_i)$. That is, c_i contains the i^{th} symbol of c_{data} and c_{ctrl}.
- **Output:** c.

Decoding algorithm: We define a decoding function $\mathrm{Dec} : [q]^n \to \{0,1\}^k \cup \{\perp\}$:

Input: A received word $z \in [q]^n$.

Output: A message $\widehat{m} \in \{0,1\}^k \cup \{\perp\}$.

Operation :

- **Unfold the control and data:** For every $i \in [n]$ we view $z_i = ((z_{\mathrm{data}})_i, (z_{\mathrm{ctrl}})_i) \in [q_{\mathrm{data}}] \times [q_{\mathrm{ctrl}}]$ and compute $z_{\mathrm{data}} = ((z_{\mathrm{data}})_1, \cdots, (z_{\mathrm{data}})_n)$ and $z_{\mathrm{ctrl}} = ((z_{\mathrm{ctrl}})_1, \cdots, (z_{\mathrm{ctrl}})_n)$.
- **Uniquely decode the control part:** $\widehat{w} = (\widehat{y}, \widehat{r}) = \mathrm{Dec}_{\mathrm{ctrl}}(z_{\mathrm{ctrl}})$.
- **List decode the data part:** Compute a list of message candidates $\mathrm{List} = \mathrm{Dec}_{\mathrm{data}}(z_{\mathrm{data}})$, by applying the list decoding for d_{data} errors.
- **Check control distance:** If $\Delta(\mathrm{Enc}_{\mathrm{ctrl}}(\widehat{w}), z_{\mathrm{ctrl}}) > d_{\mathrm{data}}$ then output $\perp$.
- **Check consistency:** Output the first message $\widehat{m} \in \mathrm{List}$ for which $h_{\widehat{r}}(\widehat{m}) = \widehat{y}$. If no such message exists output $\perp$.

4.2 Proof of Theorem 4.1.

In this section, we prove Theorem 4.1. We make use of the following technical lemma. The proof for the lemma is given below in Section 4.2.1.

Lemma 4.3. *If there are ingredients and parameters that satisfy the conditions stated in Figure 1. Moreover, if the hash family $\mathcal{H}$ specified in Figure 1 is a family of universal one-way hash functions [resp., collision resistance hash functions]. Then, the code $(\mathrm{Enc}, \mathrm{Dec})$ as specified in Figure 2 is a (q, n)-randomized code [resp., (q, n)-self-seeded code]. Moreover, the code is perfectly reliable, has alphabet size $q = q_{\mathrm{ctrl}} \cdot q_{\mathrm{data}}$, rate $R = R_{\mathrm{data}} \cdot \frac{\log(q_{\mathrm{data}})}{\log(q_{\mathrm{data}})+\log(q_{\mathrm{ctrl}})}$, and the decoding algorithm simultaneously,*

- *detects from $d_{\mathrm{ctrl}} - d_{\mathrm{data}}$ errors, and*
- *corrects from d_{data} errors.*

Proof of Theorem 4.1.

Proof of Theorem 4.1. Note that one-way functions imply universal one-way hash functions that satisfy the conditions stated for the hash family $\mathcal{H}$ in Figure 1 (see, Theorem 2.8).

Given a sufficiently small $0 < \varepsilon$, let $\hat{\varepsilon} = \varepsilon/4$ and apply Theorem 2.11 for $\hat{\varepsilon}$ and $R_{\mathrm{data}} = 1 - p - \hat{\varepsilon}$ to get a standard code $(\mathrm{Enc}_{\mathrm{data}}, \mathrm{Dec}_{\mathrm{data}})$ with rate $R_{\mathrm{data}} = 1 - p - \hat{\varepsilon}$, codeword length n_{data} alphabet size $q_{\mathrm{data}} = 2^{O(\hat{\varepsilon}^{-5})}$. Moreover, $(\mathrm{Enc}_{\mathrm{data}}, \mathrm{Dec}_{\mathrm{data}})$ list decodes from $d_{\mathrm{data}} = p \cdot n_{\mathrm{data}}$ errors (that is, from p relative errors) with a list size that depends only on $\hat{\varepsilon}$ and k. Note also that we can increase the alphabet size by bundling symbols together without damaging the list decoding and the rate R_{data} of the code, and so, we can assume that $\log(q_{\mathrm{data}})$ is of size at least $\hat{\varepsilon}^{-5}$. To summarize $(\mathrm{Enc}_{\mathrm{data}}, \mathrm{Dec}_{\mathrm{data}})$ meets the conditions stated in Figure 1 for "data code" for the parameters R_{data} and d_{data} and $\log(q_{\mathrm{data}}) > \hat{\varepsilon}^{-5}$.

Apply theorem 2.12 for $\hat{\varepsilon}$ to get a code $(\mathrm{Enc}_{\mathrm{ctrl}}, \mathrm{Dec}_{\mathrm{ctrl}})$ with codeword length $n_{\mathrm{data}} = n_{\mathrm{ctrl}}$, alphabet size at most $q_{\mathrm{ctrl}} = 2^{\hat{\varepsilon}^{-3}}$ that has distance $d_{\mathrm{ctrl}} > n_{\mathrm{ctrl}}(1 - \hat{\varepsilon})$ and efficient decoding of up to $d_{\mathrm{ctrl}}/2 - 1$ errors. Note that for a sufficiently small ε (and $\hat{\varepsilon}$) it holds that $d_{\mathrm{ctrl}} > 2d_{\mathrm{data}} + 1$. This means that $(\mathrm{Enc}_{\mathrm{ctrl}}, \mathrm{Dec}_{\mathrm{ctrl}})$ meets the conditions stated in Figure 1 for the "control code" with parameters d_{ctrl} and $\log(q_{\mathrm{ctrl}}) < \hat{\varepsilon}^{-3}$.

Thus, we can apply Lemma 4.3 for $(\mathrm{Enc}_{\mathrm{data}}, \mathrm{Dec}_{\mathrm{data}})$, $(\mathrm{Enc}_{\mathrm{ctrl}}, \mathrm{Dec}_{\mathrm{ctrl}})$ and $\mathcal{H}$ to get a (q, n)-randomized code when we take $\mathcal{H}$ to be a universal one-way hash family and (q, n)-self-seeded code if $\mathcal{H}$ is a collision resistance hash family. In both cases, it follows that

- $n = n_{\mathrm{data}}$
- $q = q_{\mathrm{data}} \cdot q_{\mathrm{ctrl}} = 2^{O(\varepsilon^{-5})}$,
- for every sufficiently small ε (and $\hat{\varepsilon}$), $R = R_{\mathrm{data}} \cdot \frac{\log(q_{\mathrm{data}})}{\log(q_{\mathrm{data}})+\log(q_{\mathrm{ctrl}})} > (1-p-\hat{\varepsilon}) \cdot \frac{1}{1+\hat{\varepsilon}} > (1-p-\varepsilon)$.
- The code detects from $d_{\mathrm{ctrl}} - d_{\mathrm{data}} = n(1 - \hat{\varepsilon}) - np > n(1 - p - \varepsilon)$ errors. That is, it detects from $1 - p - \varepsilon$ relative errors.
- The code corrects $d_{\mathrm{data}} = pn$ errors. That is, it corrects up to p relative errors.

This concludes the proof of Theorem 4.1. □

4.2.1 Proof of Lemma 4.3.

Let $\mathrm{Enc}_{\mathrm{data}}, \mathrm{Enc}_{\mathrm{ctrl}}, h_r$, be as defined in Figure 1 and let $(\mathrm{Enc}, \mathrm{Dec})$ be as defined in Figure 2. By construction, $(\mathrm{Enc}, \mathrm{Dec})$ is perfectly reliable. This holds since for every fixed randomness r if no errors are applied, all ingredients in the construction are deterministic, and standard codes are by definition perfectly reliable. Similarly, by construction $q = q_{\mathrm{ctrl}} \cdot q_{\mathrm{data}}$ and

$$R = \frac{k}{n \cdot \log(q)} = \frac{R_{\mathrm{data}} \cdot (n \cdot \log(q_{\mathrm{data}}))}{n \cdot (\log(q_{\mathrm{ctrl}}) + \log(q_{\mathrm{data}}))} = R_{\mathrm{data}} \cdot \frac{\log(q_{\mathrm{data}})}{\log(q_{\mathrm{data}}) + \log(q_{\mathrm{ctrl}})}.$$

We now prove the rest of Lemma 4.3. For simplicity, we focus on the randomized codes case, the proof for the self-seeded codes is essentially the same and follows along similar lines.[8]

Consider the following randomized experiment for a PPT algorithm A and $k \in \mathbb{N}$:

1. $(m, \mathrm{state}) \leftarrow \mathsf{A}(1^k) \in \{0,1\}^k \times \{0,1\}^*$
2. Compute $c \leftarrow \mathrm{Enc}_r(m)$ (by sampling $r \leftarrow \{0,1\}^{\ell(k)}$).
3. $z \leftarrow \mathsf{A}(m, \mathrm{state}, c)$

We make use of the following notation: Let $c_{\mathrm{data}} = \mathrm{Enc}_{\mathrm{data}}(m)$, $w = (h_r(m), r)$ and $c_{\mathrm{ctrl}} = \mathrm{Enc}_{\mathrm{ctrl}}(w)$. Recall that $z \in ([q_{\mathrm{data}}] \times [q_{\mathrm{ctrl}}])^n$, and define $z_{\mathrm{data}} = ((z_{\mathrm{data}})_1, \cdots, (z_{\mathrm{data}})_n)$ and $z_{\mathrm{ctrl}} = ((z_{\mathrm{ctrl}})_1, \cdots, (z_{\mathrm{ctrl}})_n)$ such that for every $i \in [n]$, $z_i = ((z_{\mathrm{data}})_i, (z_{\mathrm{ctrl}})_i)$. Let $\widehat{w} = (\widehat{y}, \widehat{r}) = \mathrm{Dec}_{\mathrm{ctrl}}(z_{\mathrm{ctrl}})$ and $\mathrm{List} = \mathrm{Dec}_{\mathrm{data}}(z_{\mathrm{data}})$.

Error correction: To prove the error correction property, we need to show that

$$\mathbf{Pr}\left[\Delta(c, z) \leq d_{\mathrm{data}} \wedge \mathrm{Dec}(z) \neq m\right] \leq \mathrm{neg}(k).$$

We make use of the following claims:

Claim 4.4. *If* $\Delta(c, z) \leq d_{\mathrm{data}}$ *then* $m \in \mathrm{List}$ *and* $w = \widehat{w}$.

Proof of Claim 4.4. If $\Delta(c, z) \leq d_{\mathrm{data}}$ it follows that $\Delta(c_{\mathrm{ctrl}}, z_{\mathrm{ctrl}}) \leq d_{\mathrm{data}}$ and $\Delta(c_{\mathrm{data}}, z_{\mathrm{data}}) \leq d_{\mathrm{data}}$. Since $(\mathrm{Enc}_{\mathrm{ctrl}}, \mathrm{Dec}_{\mathrm{ctrl}})$ is a standard code that uniquely decodes from $d_{\mathrm{ctrl}}/2 - 1 \geq d_{\mathrm{data}}$ errors it holds that $w = w'$. Similarly, since $(\mathrm{Enc}_{\mathrm{data}}, \mathrm{Dec}_{\mathrm{data}})$ is a standard code that list-decodes from d_{data} errors it follows that $m \in \mathrm{List}$. □

Claim 4.5. $\forall \widehat{m} \in \mathrm{List}$, *if* $\widehat{m} \neq m$ *then* $\mathbf{Pr}\left[h_r(\widehat{m}) = y\right] \leq \mathrm{neg}(k)$.

Proof of 4.5. Follows directly from the security of the universal one-way hash function, as otherwise, we can use A to find collisions in h contradicting the collision resistance property. □

By Claim 4.4, $m \in \mathrm{List}$ and $w = \widehat{w}$. Thus, by Claim 4.5, with all but negligible probability, m is the only message in List for which $h_r(m) = y$ (passing the "Check consistency" step in the decoding algorithm) which implies $\mathrm{Dec}(z) = m$.

[8] The main difference is that we consider an adaptive adversary (that is, first $r \leftarrow \{0,1\}^{\ell(k)}$ and then $(m, z) \leftarrow \mathsf{A}(r)$) and we reduce to the security of the CRH instead of the security of the UOWHF, otherwise, the proof is identical.

Error detection: To prove the error correction property, we need to show that

$$\mathbf{Pr}\left[\Delta(c,z) \leq d_{\text{ctrl}} - d_{\text{data}} \wedge \text{Dec}(z) \notin \{m, \perp\}\right] \leq \text{neg}(k).$$

We make use of the following Claim.

Claim 4.6. *If* $\Delta(c,z) \leq d_{\text{ctrl}} - d_{\text{data}}$ *and* $\text{Dec}(z) \neq \perp$ *then* $w = \widehat{w}$.

Proof. Let $\widehat{c}_{\text{ctrl}} = \text{Enc}_{\text{ctrl}}(\widehat{w})$ and note that the by the "Check control distance" step in the decoding algorithm (see Figure 2), if $\text{Dec}(z) \neq \perp$ it follows that $\Delta(z_{\text{ctrl}}, \widehat{c}_{\text{ctrl}}) \leq d_{\text{data}}$. Moreover, by construction, if $\Delta(c,z) \leq d_{\text{ctrl}} - d_{\text{data}}$ it follows that $\Delta(c_{\text{ctrl}}, z_{\text{ctrl}}) \leq d_{\text{ctrl}} - d_{\text{data}}$. Thus, if $\text{Dec}(z) \neq \perp$ and $\Delta(c,z) \leq d_{\text{ctrl}} - d_{\text{data}}$ by the triangle inequality

$$\Delta(c_{\text{ctrl}}, \widehat{c}_{\text{ctrl}}) \leq d_{\text{ctrl}}.$$

Since c_{ctrl} and $\widehat{c}_{\text{ctrl}}$ are both valid codewords of the standard code $(\text{Enc}_{\text{ctrl}}, \text{Dec}_{\text{ctrl}})$ that has distance d_{ctrl}, it follows that $w = \widehat{w}$. □

By Claim 4.6 if $\text{Dec}(z) \neq \perp$ then $w = \widehat{w}$. This concludes the proof, since by Claim 4.5, if $w = w'$, with all but negligible probability, m is the only valid message that can be outputted by $\text{Dec}(z)$.

5 Randomized Codes Imply One Way Functions.

We now show that non-trivial randomized codes imply the existence of one-way functions. For large alphabets, this is indeed optimal since as we have seen in our construction it is possible to construct optimal randomized codes assuming only universal one-way functions (which can be constructed from one-way functions).

The following theorem states that if a randomized code has a rate that bypasses the singleton bound, it implies one-way functions.

Theorem 5.1 (Randomized codes with rate beating the Singleton bound imply one way functions)**.** *Let* (Enc, Dec) *be a randomized code with rate* $0 \leq R < 1$. *If one of the following holds:*

- *The code detects* ρ_0*-relative errors and* $\rho_0 > 1 - R + \frac{1}{n}$.
- *The code corrects from* ρ_1*-relative errors and* $2\rho_1 > 1 - R + \frac{2}{n}$.

Then there exists one-way functions.

The proof will make use of the following lemma.

Lemma 5.2 (Non-trivial error detection/correction for randomized codes implies one-way functions)**.** *Let* (Enc, Dec) *be an* (n,q)*-randomized code that has at least one of the following properties:*

- *The code detects* d*-errors.*
- *The code corrects* $\lceil d/2 \rceil$*-errors.*

Let $s := n - d$, *if* $v := \lceil \log(q) \cdot s \rceil \leq k - 1$ *then* $\mathsf{F}\colon \{0,1\}^{k+\ell} \to \{0,1\}^v$ *defined so that on input* $(m,r) \in \{0,1\}^{k+\ell}$ *it computes* $\text{Enc}_r(m)$ *and outputs a binary representation of the first* s *symbols is a distributional one-way function.*[9] [10]

The proof for the lemma is given below, but first, we will use it to prove Theorem 5.1.

[9] Here, ℓ is just the implicitly assumed bound on the number of random bits used by Enc.

[10] The statement of Lemma 5.2 could be readily extended so that restricting the output of $\text{Enc}_r(m)$ to *any* s symbols (that can be

Proof of Theorem 5.1.

Proof of Theorem 5.1. Let $d = Max(\lceil \rho_0 \cdot n \rceil, \lceil 2\rho_1 \cdot n \rceil)$ and note that by definition at least one of the following holds: The code detects d-errors, or corrects $\lceil d/2 \rceil$-errors. Moreover, by our assumption on the rate of the code, it holds that $d > n - R \cdot n + 1$. Let $s = n - d$ and let $v = \lceil \log(q) \cdot s \rceil$ since $R \geq \frac{k}{\log(q) \cdot n}$ it holds that $v \leq k - 1$. Thus, we meet the conditions of Lemma 5.2 which implies that there exists a distributional one-way function (in fact outputting the binary representation of the first s symbols of Enc is the distributional one-way function), and by Theorem 2.6 there exists a one-way function. □

Proof of Lemma 5.2.

Proof of Lemma 5.2. For $k \in \mathbb{N}$, let M_k and R_k denote the random variables sampled uniformly from $\{0,1\}^k$ and $\{0,1\}^{\ell(k)}$ respectively. Let $Y_k = \mathsf{F}(M_k, R_k)$ and assume for contradiction that F is not a distributional one-way function (see Definition 2.5). That is, for every constant $c > 0$ there exists a PPT algorithm A such that for infinitely many $k \in \mathbb{N}$,

$$\mathsf{SD}\Big(\big(\mathsf{A}(Y_k), Y_k\big), \big((M_k, R_k), Y_k\big)\Big) \leq 1/k^c. \tag{1}$$

For a fixed k for which the above holds, we omit the subscript k from the random variables to avoid clutter.

Let $(M', R') = \mathsf{A}(Y)$ and note that by the definition of average min-entropy (see Definition 2.2)

$$\mathbf{Pr}\left[M = M'\right] \leq 2^{-\tilde{H}_\infty(M|Y)} \leq 1/2 \tag{2}$$

where the last inequality follows by Lemma 2.3 that states that $\tilde{H}_\infty(M|Y) \geq H_\infty(M, Y) - v \geq k - v \geq 1$.

Let $C = \mathrm{Enc}_R(M)$ and $C' = \mathrm{Enc}_{R'}(M')$ and recall that by definition, $Y = \mathsf{F}(M, R)$ is just the binary representation of the first s symbols of C. Thus, if $\mathsf{F}(M', R') = \mathsf{F}(M, R) = Y$ it follows that, $\Delta(\mathrm{Enc}_R(M), \mathrm{Enc}_{R'}(M')) \leq n - s = d$, which by Equation 1 implies that:

$$\mathbf{Pr}\left[\Delta(C, C') > d\right] \leq 1/k^c. \tag{3}$$

Note also that by Equation 1,

$$\mathbf{Pr}\left[M' = \perp\right] \leq 1/k^c \tag{4}$$

We first handle the case where the code detects d-error and then the case where the code corrects $\lceil d/2 \rceil$-errors.

Detection: By Equation 1 and by the reliability of the code,

$$\mathbf{Pr}\left[\mathrm{Dec}(C') = M'\right] \geq \mathbf{Pr}\left[\mathrm{Dec}(C) = M\right] - 1/k^c > 0.9 \tag{5}$$

We now define a PPT algorithm A' (that makes use of A) that contradicts the detection property of the code: On input 1^k, A' samples $m \leftarrow \{0,1\}^k$ and on input $c \in [q]^n$ it runs $(m', r') \leftarrow \mathsf{A}(y)$ where y is just the binary representation of the first s symbols of c (i.e., $y = \mathsf{F}(m, r)$), and outputs $c' = \mathrm{Enc}_{r'}(m')$. It holds that:

efficiently identified) not just the first s symbols is itself a distributional one-way function. This would be similar to our statement on self-seeded codes made at Lemma 6.2, where we show that restricting the output to any subset of size s is a collision resistance hash family. However, for the sake of simplicity, we restrict our statement and analysis to the simple case where the distributional one-way function is defined to be the first s symbols.

- $M = \mathsf{A}'(1^k)$ (where by definition M is sampled uniformly by A').
- Compute $C = \mathrm{Enc}_R(M)$ (by sampling $R \leftarrow \{0,1\}^\ell$).
- $C' = \mathrm{Enc}_{R'}(M') = \mathsf{A}'(M, C)$.

Thus, A' contradicts the d-detection property since

$$\begin{aligned}
&\mathbf{Pr}\left[\mathrm{Dec}(C') \notin \{M, \perp\} \wedge \Delta(C', C) \leq d\right] \\
&\geq \mathbf{Pr}\left[\mathrm{Dec}(C') = M' \wedge M' \notin \{M, \perp\} \wedge \Delta(C', C) \leq d\right] \\
&\geq \mathbf{Pr}\left[\mathrm{Dec}(C') = M'\right] - \mathbf{Pr}\left[M' = M\right] - \mathbf{Pr}\left[M' = \perp\right] - \mathbf{Pr}\left[\Delta(C', C) > d\right] \\
&\geq 0.9 - 0.5 - 2/k^c \geq 0.1
\end{aligned}$$

Where the penultimate inequality follows from Equations 5, 2, 4 and 3.

Correction: We now prove the correction property. Let G be a function defined as follows: On inputs $c, c' \in [q]^n$, G samples a set I of random $(n-s)/2$ indices from the set $[n] \setminus [s]$ (if $n - s$ is an odd number then with probability $1/2$ it sample $\lceil (n-s)/2 \rceil$ indices and with probability $1/2$ it samples $\lceil (n-s)/2 \rceil - 1$ indices), and outputs $c^* \in [q]^n$ where for every $i \in I$, $c^*_i = c'_i$ and for every $i \in [n] \setminus I$, $c^*_i = c_i$. Intuitively, if c and c' agree on the first s indexes then $G(c, c') = c^*$ is a random "mid point" between c and c'.

Recall that $C = \mathrm{Enc}_R(M)$, $C' = \mathrm{Enc}_{R'}(M')$ and let $C^* = G(C, C')$. By Equation 1 and symmetry,

$$\mathsf{SD}((M, R, C, C^*), (M', R', C', C^*)) \leq 2/k^c. \tag{6}$$

Since by assumption $n - s = d$, by the definition of C^* it follows that:

$$\mathbf{Pr}\left[\Delta(C, C^*) > \lceil d/2 \rceil\right] \leq 1/k^c \tag{7}$$

Recall that (M, R) are sampled uniformly at random, thus, by Equations 6 and 7 and the decoding properly of the code

$$\mathbf{Pr}\left[\mathrm{Dec}(C^*) = M'\right] \geq \mathbf{Pr}\left[\mathrm{Dec}(C^*) = M\right] - 3/k^c \geq 0.9. \tag{8}$$

We now define a PPT algorithm A' (that makes use of A) that contradicts the correction property of the code: On input 1^k, A' samples $m \leftarrow \{0,1\}^k$ and on input $c \in [q]^n$ runs $(m', r') \leftarrow \mathsf{A}(y)$ where y is just the binary representation of the first s symbols of c (i.e., $y = \mathsf{F}(m, r)$), computes $c' = \mathrm{Enc}_{r'}(m')$ and outputs $c^* = G(c, c')$. It holds that,

- $M = \mathsf{A}'(1^k)$ (where by definition M is sampled uniformly by A').
- Compute $C = \mathrm{Enc}_R(M)$ (by sampling $R \leftarrow \{0,1\}^\ell$).
- $C^* = \mathsf{A}'(M, C)$ (recall that $C^* = G(C, C')$ for $C' = \mathrm{Enc}_{R'}(M')$ and $(M', R') = \mathsf{A}(\mathsf{F}(M, R))$).

Thus, A' contradicts the $\lceil d/2 \rceil$-correction property since,

$$\begin{aligned}
&\mathbf{Pr}\left[\mathrm{Dec}(C^*) \neq M \wedge \Delta(C^*, C) \leq \lceil d/2 \rceil\right] \\
&\geq \mathbf{Pr}\left[\mathrm{Dec}(C^*) = M' \wedge M \neq M' \wedge \Delta(C^*, C) \leq \lceil d/2 \rceil\right] \\
&\geq \mathbf{Pr}\left[\mathrm{Dec}(C^*) = M'\right] - \mathbf{Pr}\left[M = M'\right] - \mathbf{Pr}\left[\Delta(C^*, C) \leq \lceil d/2 \rceil\right] \\
&\geq 0.9 - 0.5 - 1/k^c \geq 0.1.
\end{aligned}$$

Where the penultimate inequality follows from Equations 2, 7 and 8. This concludes the proof for Lemma 5.2.

□

6 Self-Seeded Codes Imply Collision Resistance Hash Functions.

We now show that non-trivial self-seeded codes imply the existence of a collision resistance hash family. Loosely speaking, the following theorem states that if a self-seeded code has rate that bypasses the Singleton bound, then it implies collision resistance hash functions (since there are codes with large alphabet that can achieve the Singleton bound, this result means that for large alphabet, any non-trivial self-seeded code implies collision resistance functions).

We note that while the result below is stated with respect to the notion of self-seeded codes, the same exact proof also holds for the weaker notion seeded codes defined in [GHY20] in which the encoding and decoding algorithms both share the random seed. Since our lower bound still holds even if the decoding algorithm is given the random seed as input, the lower bound immediately extends to the weaker notion of seeded codes. However, to avoid confusion and clutter, we present our result with respect to the notion of self-seeded codes defined in this paper.

Theorem 6.1 (Self-Seeded codes with rate beating the Singleton bound imply collision resistance hash functions)**.** *Let* $(\mathrm{Enc}, \mathrm{Dec})$ *be a self-seeded code with rate* $0 \leq R < 1$*. If one of the following holds:*

- *The code detects* ρ_0*-relative errors and* $\rho_0 > 1 - R + \frac{1}{n}$*.*
- *The code corrects from* ρ_1*-relative errors and* $2\rho_1 > 1 - R + \frac{2}{n}$*.*

Then there exists an explicit construction for a collision resistance hash family given black-box access to the code.

We mention that while the above theorem states that we can construct collision resistance hash functions from non-trivial self-seeded codes. In fact, given a non-trivial self-seeded code, restricting the output of the encoding to *any* sufficiently large subset (that can be efficiently identified) of symbols in a way that is still shrinking (that is, the output of the restricted part is less than the input) is itself a collision resistance hash function. This is stated formally in the following lemma.

Lemma 6.2 (Non-trivial error detection/correction for self-seeded codes implies collision resistance functions)**.** *Let* $(\mathrm{Enc}, \mathrm{Dec})$ *be an* (n, q)*-self-seeded code that has at least one of the following properties:*

- *The code detects* d*-errors.*
- *The code corrects* $\lceil d/2 \rceil$*-errors.*

And let T *be a* PPT *algorithm that on input* 1^k *outputs a set* $t \subseteq [n]$ *of size* s *such that* $n - d \leq s$ *and* $v := \lceil \log(q) \cdot s \rceil \leq k - 1$*. Then the family* $\mathcal{H} = \{\mathrm{Enc}_r|_{\mathsf{T}} \colon \{0,1\}^k \to \{0,1\}^v\}_{r \in \{0,1\}^\ell}$*, is collision resistance, where we define* $\mathrm{Enc}_r|_{\mathsf{T}}$ *such that for every* $m \in k$*,* $\mathrm{Enc}_r|_{\mathsf{T}}(m)$ *outputs the ordered sequence of* $(\mathrm{Enc}_r(m))_{i \in t}$ *as a binary string of size* v*.*[11]

We prove Lemma 6.2 below, but first we will use it to prove Theorem 6.1.

Proof of Theorem 6.1.

Proof of Theorem 6.1. Let $d = Max(\lceil \rho_0 \cdot n \rceil, \lceil 2\rho_1 \cdot n \rceil)$ and note that by definition at least one of the following holds: The code detects d-errors, or corrects $\lceil d/2 \rceil$-errors. Moreover, by our assumption on the rate of the code, it holds that $d > n - R \cdot n + 1$. Let $s = n - d$ and let $v = \lceil \log(q) \cdot s \rceil$ since $R \geq \frac{k}{\log(q) \cdot n}$ it

[11] $\mathrm{Enc}_r|_{\mathsf{T}}$ is just the restriction of Enc_r to the indexes specified by T, where we write the output as a binary string. Moreover, ℓ is just the bound on the number of random uniform coins used by the self-seeded code.

holds that $v \leq k-1$. Let T be a PPT algorithm that outputs the set $[s]$ (that is, it outputs the first s indexes in n). To conclude, note that T and $(\mathrm{Enc}, \mathrm{Dec})$ satisfy the conditions of Lemma 6.2, which implies that there exists a collision resistance hash family (in fact, outputting the binary representation of the s symbols in Enc that are specified by T is the hash function).

□

Proof of Lemma 6.2.

Proof of Lemma 6.2. By assumption $v \leq k-1$ which implies that family $\mathcal{H}$ is shrinking. Moreover, since T and Enc are PPT algorithms it follows the for every $r \in \{0,1\}^{\ell}$ and every $m \in \{0,1\}^{k}$, $\mathrm{Enc}_r|_{\mathsf{T}}(m)$ can be evaluated in $\mathrm{poly}(k)$. Thus, to conclude the proof it remains to prove the collision resistance property. Assume towards contradiction that this does not hold. That is, there exists a PPT A and $p \in \mathrm{poly}$ such that for infinitely many $k \in \mathbb{N}$,

$$\Pr_{\substack{r \leftarrow \{0,1\}^{\ell(k)}, \\ m,m' \leftarrow \mathsf{A}(r)}} \left[m \neq m' \wedge \mathrm{Enc}_r|_{\mathsf{T}}(m) = \mathrm{Enc}_r|_{\mathsf{T}}(m') \right] \geq p(k).$$

Fix a $k \in \mathbb{N}$ for which the above holds. We now show that A can be used to contradict the d-detection and the $\lceil d/2 \rceil$-correction properties of the code.

Detection: Consider the PPT adversary A' that on input $r \in \{0,1\}^{\ell}$, computes $(m, m') \leftarrow \mathsf{A}(r)$ and outputs $(m, z := \mathrm{Enc}_r(m'))$. Recall that by definition, for every $m, m' \in \{0,1\}^{k}$ and every $r \in \{0,1\}^{\ell}$, if $\mathrm{Enc}_r|_{\mathsf{T}}(m) = \mathrm{Enc}_r|_{\mathsf{T}}(m')$ then it implies that, $\Delta(\mathrm{Enc}_r(m), \mathrm{Enc}_r(m')) \leq n - s \leq d$ (where the inequality follows, by our assumption that $n - d \leq s$). Thus, it follows that:

$$\begin{aligned}
&\Pr_{\substack{r \leftarrow \{0,1\}^{\ell}, \\ m,z \leftarrow \mathsf{A}'(r)}} \left[\Delta(\mathrm{Enc}_r(m), z) \leq d \wedge \mathrm{Dec}(z) \notin \{\perp, m\} \right] \\
&\geq \Pr_{\substack{r \leftarrow \{0,1\}^{\ell}, \\ m,z \leftarrow \mathsf{A}'(r)}} \left[m \neq m' \wedge \Delta(\mathrm{Enc}_r(m), z) \leq d \wedge \mathrm{Dec}(z) = m' \right] \\
&\geq p(k) - \mathrm{neg}(k).
\end{aligned}$$

Where the last inequality follows since the code is reliable (which implies that $\mathbf{Pr}\left[\mathrm{Dec}(z) = m'\right] \geq 1 - \mathrm{neg}(k)$).

We now argue about the correction property of the code:

Correction: Consider the PPT adversary A' that on input $r \in \{0,1\}^{\ell}$, computes $(m, m') \leftarrow \mathsf{A}(r)$ it computes $\mathrm{Enc}_r(m)$ and $\mathrm{Enc}_r(m')$ and outputs (m, z) such that $\Delta(z, \mathrm{Enc}_r(m)) \leq \lceil d \rceil$ and $\Delta(z, \mathrm{Enc}_r(m')) \leq \lceil d \rceil$ and note that this is possible when $\Delta(\mathrm{Enc}_r(m), \mathrm{Enc}_r(m')) \leq d$ otherwise output an arbitrary string. Recall that similar to the detection analysis, for every $m, m' \in \{0,1\}^{k}$ and every $r \in \{0,1\}^{\ell}$, if $\mathrm{Enc}_r|_{\mathsf{T}}(m) = \mathrm{Enc}_r|_{\mathsf{T}}(m')$ then it implies that, $\Delta(\mathrm{Enc}_r(m), \mathrm{Enc}_r(m')) \leq n - s \leq d$.

Thus, it follows that:

$$\begin{aligned}
&\Pr_{\substack{r \leftarrow \{0,1\}^{\ell}, \\ m,z \leftarrow \mathsf{A}'(r)}} \left[\Delta(\mathrm{Enc}_r(m), z) \leq \lceil d/2 \rceil \wedge \mathrm{Dec}(z) \neq m \right] \\
&\geq \Pr_{\substack{r \leftarrow \{0,1\}^{\ell}, \\ m,z \leftarrow \mathsf{A}'(r)}} \left[m \neq m' \wedge \Delta(\mathrm{Enc}_r(m), z) \leq \lceil d/2 \rceil \wedge \Delta(\mathrm{Enc}_r(m'), z) \leq \lceil d/2 \rceil \wedge \mathrm{Dec}(z) = m' \right] \\
&\geq p(k) - \mathrm{neg}(k).
\end{aligned}$$

Where the last inequality follows by the decoding property of the code, which implies that $\mathbf{Pr}\left[\Delta(\mathrm{Enc}_r(m'), z) \leq \lceil d/2 \rceil \wedge \mathrm{Dec}(z) = m'\right] \geq 1 - \mathrm{neg}(k)$. This concludes the proof of Lemma 6.2. □

References

[BGGZ21] Jeremiah Blocki, Venkata Gandikota, Elena Grigorescu, and Samson Zhou. Relaxed locally correctable codes in computationally bounded channels. *IEEE Transactions on Information Theory*, 67(7):4338–4360, 2021.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM journal on computing*, 38(1):97–139, 2008.

[GHY20] Ofer Grossman, Justin Holmgren, and Eylon Yogev. Transparent error correcting in a computationally bounded world. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part III*, volume 12552 of *Lecture Notes in Computer Science*, pages 530–549. Springer, 2020.

[GR08] Venkatesan Guruswami and Atri Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Transactions on information theory*, 54(1):135–150, 2008.

[GRS] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. Essential coding theory.

[GS16] V. Guruswami and A. Smith. Optimal rate code constructions for computationally simple channels. *Journal of the ACM (JACM)*, 63(4):35, 2016.

[Ham50] R. W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, 1950.

[HOSW11] Brett Hemenway, Rafail Ostrovsky, Martin J. Strauss, and Mary Wootters. Public key locally decodable codes with short keys. In Leslie Ann Goldberg, Klaus Jansen, R. Ravi, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 14th International Workshop, APPROX 2011, and 15th International Workshop, RANDOM 2011, Princeton, NJ, USA, August 17-19, 2011. Proceedings*, volume 6845 of *Lecture Notes in Computer Science*, pages 605–615. Springer, 2011.

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography. In *FOCS 30*, pages 230–235, 1989.

[KMRZS17] Swastik Kopparty, Or Meir, Noga Ron-Zewi, and Shubhangi Saraf. High-rate locally correctable and locally testable codes with sub-polynomial query complexity. *Journal of the ACM (JACM)*, 64(2):1–42, 2017.

[Lip94] R. J. Lipton. A new approach to information theory. In *11th Annual Symposium on Theoretical Aspects of Computer Science*, pages 699–708, 1994.

[MPSW10] Silvio Micali, Chris Peikert, Madhu Sudan, and David A. Wilson. Optimal error correction for computationally bounded noise. *IEEE Trans. Information Theory*, 56(11):5673–5680, 2010.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 33–43. ACM, 1989.

[OPS07] Rafail Ostrovsky, Omkant Pandey, and Amit Sahai. Private locally decodable codes. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, volume 4596 of *Lecture Notes in Computer Science*, pages 387–398. Springer, 2007.

[Rey11] Leonid Reyzin. Some notions of entropy for cryptography: (invited talk). In *International Conference on Information Theoretic Security*, pages 138–142. Springer, 2011.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 387–394. ACM, 1990.

[Sha48] Claude E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27(4):623–656, 1948.

[SS21a] R. Shaltiel and J. Silbak. Explicit list-decodable codes with optimal rate for computationally bounded channels. *Comput. Complex.*, 30(1):3, 2021.

[SS21b] R. Shaltiel and J. Silbak. Explicit uniquely decodable codes for space bounded channels that achieve list-decoding capacity. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1516–1526, 2021.

[SS22] R. Shaltiel and J. Silbak. Error correcting codes that achieve BSC capacity against channels that are poly-size circuits. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 13–23, 2022.

[SS24] R. Shaltiel and J. Silbak. Explicit codes for poly-size circuits and functions that are hard to sample on low entropy distributions. *Electronic Colloquium on Computational Complexity (ECCC)*, 2024.