Improved Pseudorandom Codes from Permuted Puzzles

Miranda Christ
Columbia University
mchrist@cs.columbia.edu

Noah Golowich
Microsoft Research
noah.golowich@austin.utexas.edu

Sam Gunn
UC Berkeley
gunn@berkeley.edu

Ankur Moitra
MIT
moitra@mit.edu

Daniel Wichs
Northeastern University
NTT Research
wichs@ccs.neu.edu

Abstract

Watermarks are an essential tool for identifying AI-generated content. Recently, Christ and Gunn (CRYPTO '24) introduced *pseudorandom error-correcting codes* (PRCs), which are equivalent to watermarks with strong robustness and quality guarantees. A PRC is a pseudorandom encryption scheme whose decryption algorithm tolerates a high rate of errors. Pseudorandomness ensures quality preservation of the watermark, and error tolerance of decryption translates to the watermark's ability to withstand modification of the content.

In the short time since the introduction of PRCs, several works (NeurIPS '24, RANDOM '25, STOC '25) have proposed new constructions. Curiously, all of these constructions are vulnerable to quasipolynomial-time distinguishing attacks. Furthermore, all lack robustness to edits over a constant-sized alphabet, which is necessary for a meaningfully robust LLM watermark. Lastly, they lack robustness to adversaries who know the watermarking detection key. Until now, it was not clear whether any of these properties was achievable individually, let alone together.

We construct pseudorandom codes that achieve all of the above: plausible subexponential pseudorandomness security, robustness to worst-case edits over a binary alphabet, and robustness against even computationally unbounded adversaries that have the detection key. Pseudorandomness rests on a new assumption that we formalize, the *permuted codes conjecture*, which states that a distribution of permuted noisy codewords is pseudorandom. We show that this conjecture is implied by the permuted puzzles conjecture used previously to construct doubly efficient private information retrieval. To give further evidence, we show that the conjecture holds against a broad class of simple distinguishers, including read-once branching programs.

Contents

1 Introduction **1**
- 1.1 Results . . . 2
- 1.2 Related work . . . 5

2 Preliminaries **7**
- 2.1 Pseudorandom codes . . . 7
- 2.2 Watermarking schemes . . . 8

3 Technical overview **8**
- 3.1 The computational assumption: permuted codes . . . 9
- 3.2 Edit-robust pseudorandom codes from permuted codes assumption . . . 12
- 3.3 Edit-robust watermarking schemes using folded Reed-Solomon codes . . . 14

4 The permuted codes assumption **16**
- 4.1 Comparison to permuted puzzles . . . 16
- 4.2 Statistical evidence for small alphabets . . . 18
- 4.3 The alphabet permutation is necessary . . . 21
- 4.4 A quasipolynomial-time distinguisher for a natural class of constructions . . . 23

5 Edit-robust pseudorandom codes **25**
- 5.1 Preliminaries on folded Reed-Solomon codes . . . 25
- 5.2 Preliminaries on edit distance . . . 26
- 5.3 Our PRC construction: Algorithm 1 . . . 27
- 5.4 Analysis of Algorithm 1 . . . 30
- 5.5 From PRCs to watermarking: background . . . 36
- 5.6 From PRC to watermarking: soundness, undetectability, & robustness . . . 37

6 Substitution-robust pseudorandom codes **40**
- 6.1 Proof of Proposition 6.1 . . . 42

1 Introduction

The proliferation of AI-generated content creates serious challenges for society today, due to increasingly realistic threats such as misinformation and impersonation. One way to address this challenge is to use *watermarking*, which involves embedding a hidden signal in generated content that allows it to be detected as such. This content can later be passed to a *watermarking detector* which can identify the content's provenance using the hidden signal. In order to be useful, watermarking schemes need to satisfy certain properties: first, they should not significantly degrade the quality of the content, and second, they should be robust to modifications of the content by an adversary (e.g., one seeking to remove the watermark). In recent years, a promising approach to watermarking which comes with provable guarantees and also has shown empirical promise [GZS25, YZC+25, HLLL25] is that of pseudorandom codes [CG24]. The goal of this paper is to provide pseudorandom codes, and thereby watermarking schemes, with improved quality and robustness guarantees.

A *pseudorandom code (PRC)* is a secret-key encryption scheme that is both *pseudorandom*, in that any polynomial number of ciphertexts are computationally indistinguishable from uniform, and *error-correcting* or *robust*, in that decryption succeeds even when ciphertexts are passed through some error channel. [CG24] showed a direct way to construct a watermark for essentially any kind of AI-generated content from any pseudorandom code. Error correction of the PRC implies that the watermark persists under content modifications; pseudorandomness translates to *undetectability*, meaning that the output of the watermarking scheme is *computationally indistinguishable* from that of the true language model, which provides a very strong guarantee against degradations in quality. In fact, a PRC may be viewed as a special case of a watermarking scheme, meaning that PRCs are equivalent to watermarking schemes in an appropriate sense.

In the short time since PRCs were introduced, there has been a significant body of work building them [GM24, GG25, AAC+25], applying them to watermarks [GZS25, CHS25, GM24], and investigating their relationship to other cryptographic primitives [GGW25, DMR25]. An important direction is to construct PRCs with stronger pseudorandomness and robustness, which would immediately yield improved watermarks. For example, the first PRC proposed in [CG24] withstood only *substitution* errors or *random deletions*, resulting in an LLM watermark that tolerates word substitutions and random deletions. Ideally, a fully satisfactory watermark should be robust to *non-random edits* as well, and [GM24] later built one by constructing the first edit-robust PRC, although it required an unrealistic assumption on the entropy of generated text. We remark that our focus in this paper is on watermarking schemes for *language models*, for which edits are a particularly natural notion by which to consider modification of content. Nevertheless, PRCs have been shown to be useful for other types of models as well, such as image diffusion models [GZS25], and our results have implications in those domains as well.

Despite the intense interest in the area, we still lack PRCs with *any one of* the following three basic and desirable properties:

Property (1): subexponential security. Until now, all PRCs suffered from quasipolynomial-time attacks against pseudorandomness. Arguably, the biggest open question surrounding PRCs was whether it was possible to achieve subexponential security at all. The basic issue was that previous constructions all involved hiding a $(\log n)$-sparse structure in a length-n codeword. This sparsity was crucial for error correction; any larger structure would be destroyed by errors. But the price of using such sparse patterns was that there necessarily existed $n^{\log n}$-time brute-force search

distinguishing attacks. Therefore, stronger pseudorandomness seemed to call for new techniques, or possibly even be unattainable.

Property (2): small-alphabet edit robustness. LLM watermarks should withstand edits to the text, and doing so requires edit-robust PRCs. While [GM24] constructed an edit-robust PRC, theirs used a polynomial-size alphabet. This translates to a strong entropy requirement for their watermark: It can only be detected if there is a large amount of entropy per word in the text. Unfortunately, realistic text has only about one bit of entropy per word, necessitating an edit-robust PRC with a *constant-sized*, ideally binary, alphabet.

Property (3): strongly-adaptive robustness. Previous works on PRCs considered error channels with varying levels of power. The weakest are random errors, and the strongest error models from prior work were computationally bounded channels that have some partial information about the decoding key [AAC+25]. Their corresponding watermarks withstand attackers with similar levels of power—in the case of [AAC+25], the attacker can query a watermarked model and the watermark detector. However, an attacker that knows the detection key can easily find much smaller perturbations that remove the watermark. In fact, as a consequence of the weak pseudorandomness of existing PRCs, it is often possible for motivated attackers to learn the detection key for practical parameter settings.[1]

Therefore, we would ideally like to ensure robustness even when the attacker has *full knowledge of the watermarking key*.[2] Such a guarantee, which we call "strongly-adaptive robustness" and which corresponds to the worst-case error model typically considered in coding theory, remained open even for robustness to substitution errors prior to this work. This is not merely an academic goal but has important implications for how watermarking schemes can be deployed, since it would allow anyone to verify a watermark rather than just one trusted party, without compromising robustness.

1.1 Results

In this work, ***we present a new PRC construction, and accompanying watermarking scheme, satisfying all three of the above properties.*** In order to prove pseudorandomness, some computational hardness assumption is necessary. The pseudorandomness of our PRCs is based on the *permuted codes conjecture*, which states that the following distribution is pseudorandom for certain codes $C \subseteq \mathbb{F}_q^n$ and any noise rate $\eta = \Omega(1)$:

1. Sample an index permutation π over $[n]$ and alphabet permutations $\pi_1, \ldots, \pi_n$ over $[q]$.
2. Sample codewords $c_1, \ldots, c_T$ uniformly at random from C.
3. For each codeword $c_i = (c_{i,1}, \ldots, c_{i,n})$, permute the location indices according to π and the alphabet at each location index j according to π_j, obtaining $\hat{c}_i = (\pi_1(c_{i,\pi(1)}), \ldots, \pi_n(c_{i,\pi(n)}))$.

[1] In the most robust image watermark used by [GZS25], the PRC has parity checks of size 3, resulting in an n^3-time key recovery attack: an attacker can iterate over all n^3 possible parity checks and test which are correlated with the watermarked images. See the "Practical undetectability" section of [GZS25] for discussion on their choice of parameters.

[2] An astute reader may be concerned that an attacker that recovers the detection key may also be able to embed watermarks in any content of their choice. Prior works have constructed watermarks that are unforgeable even by an adversary that knows the detection key [CG24, FGJ+24, LSS26].

4. Output $\hat{c}_1+e_1, \ldots, \hat{c}_T+e_T$, where the error $e_i = (e_{i,1}, \ldots, e_{i,n})$ is sampled with each coordinate $e_{i,j}$ being 0 with probability $1-\eta$ and uniformly random from $\mathbb{F}_q$ otherwise.

Conjecture (Permuted codes conjecture, Conjecture 3.1)**.** *Let C be any family of linear codes over $\mathbb{F}_q$ with dual distance $n^{\Omega(1)}$ and let $\eta \in (0,1)$ be a constant rate of random substitutions. The* permuted codes conjecture *says that the above distribution is pseudorandom for any $T = \text{poly}(n)$.*

We state the general form of the conjecture, since we are unable to find any counterexamples, and wish to provide a broad target for cryptanalysis. However, for our applications, we will only require the conjecture to hold for specific codes C. The conjecture also plausibly holds with even sub-exponential security (i.e., for some constant $c > 0$ and any number of samples $T = 2^{O(n^c)}$, attackers running in time $O(2^{n^c})$ have at most $2^{-\omega(n^c)}$ advantage in distinguishing the above permuted code distribution from uniform). This translates to plausible sub-exponential security for undetectability in our watermarking applications. While Conjecture 3.1 has not been stated before in its present form, we show that it follows from the "permuted puzzles" assumption of [BHMW21, BW21] which was originally used to obtain doubly efficient private information retrieval:

Theorem (Evidence for the permuted codes conjecture, Proposition 3.4)**.** *The "permuted puzzles" conjecture [BHMW21, BW21] implies the permuted codes conjecture.*

We note that earlier versions of the permuted puzzles conjecture were introduced by [BIPW17, CHR17] and further studied in [BHW19]. We focus on the variants of [BHMW21, BW21], because we are able to show they imply the permuted codes conjecture.

As the permuted codes and permuted puzzles conjectures are relatively new, we perform some cryptanalysis to better our understanding. On the positive side, we show statistical evidence for the permuted codes conjecture over constant-size alphabets: A sample of $O(\log n)$ codewords are jointly statistically uniform (for comparison recall that Conjecture 3.1 posits that $\text{poly}(n)$ codewords are computationally indistinguishable from uniform).

Theorem (Statistical uniformity of a few codewords, Corollary 4.3)**.** *Let C be any code with polynomial dual distance over a constant-sized alphabet. Then there exists a $T = \Omega(\log n)$ such that T samples from C's permuted code distribution (per Conjecture 3.1) are statistically indistinguishable from T uniformly random strings.*

There are some conceptual similarities between the Permuted Codes conjecture and the "low-degree conjecture," which has been extensively studied in the context of algorithmic statistics in recent years (e.g., [Hop18, KWB19, BHJK25], amongst many others). Roughly speaking, the low-degree conjecture states that for any distribution P which is permutation invariant and indistinguishable from uniform by low-degree polynomials (which mirrors our requirement of "high dual distance"), there is *no efficient algorithm* distinguishing P from uniform. We observe that Corollary 4.3 in fact establishes a special case of the low-degree conjecture by showing that the distribution P is statistically close to uniform; see Corollary 4.4.[3]

On the negative side, we consider a modification of the permuted codes conjecture in which one omits the alphabet permutations applied to the individual symbols of the codewords. We show in Theorem 4.5 that the conjecture would be false in general in this case, by taking C to be the Reed-Solomon code. In particular, we show that *even a constant number of codewords would be efficiently*

[3]Incidentally, recent independent work [BHJK25] has established that the low-degree conjecture is false more generally.

Randomization method	Secure?	Explanation
Symbol + alphabet permutations	✗	This is the "toy conjecture" of [BIPW17], which was shown to be insecure when instantiated with Reed–Solomon codes in [BHMW21, BW21].
Noise + index permutation	✗	We prove in Theorem 4.5 that this is insecure when instantiated with Reed–Solomon codes.
Noise + alphabet permutations	✗	Insecure for any efficiently decodable binary linear code: Over $\mathbb{F}_2$, the alphabet permutations are simply a one-time pad, which can be removed by adding pairs of samples together (though this roughly doubles the noise rate). Any efficient decoder then serves as a distinguisher.
Noise + index + alphabet permutations	✓	This is either our permuted codes conjecture or the permuted puzzles conjecture, depending on whether the noise is substitutions or erasures.

Table 1: Any two of the three randomizing methods employed in the permuted codes and puzzles conjectures are insufficient to guarantee pseudorandomness in general. As in the permuted codes assumption, we consider codes with polynomial dual distance.

distinguishable from random. In fact, it turns out that all three of the randomizations (alphabet permutation, index permutation, and noise) are necessary: The permuted codes conjecture would be false if any one of these was omitted. See Table 1 for more details.

Improved PRCs and watermarking schemes from permuted codes. The permuted codes conjecture for any specific efficiently (list-)decodable code $C \subseteq \mathbb{F}_q^n$ directly gives PRCs with strong adaptive robustness against *substitutions* over the alphabet $\mathbb{F}_q$. Essentially the PRC ciphertexts are the outputs of the permuted codeword distribution, and the detection procedure undoes the permutations and applies the decoding algorithm of the underlying code. The rate of substitutions that the PRC tolerates can be set arbitrarily close to the error tolerance of the underlying code C. See Section 6 for this basic result.

The basic result only handles substitutions. As our main positive result for PRCs, we show how to also handle *edits*. Under the permuted codes conjecture, we can obtain PRCs satisfying all three of the desired properties from Section 1 above:

Theorem (An improved PRC from permuted codes, Theorem 5.6)**.** *Under the permuted codes conjecture, there exists a binary-alphabet PRC that is strongly-adaptive to some constant rate of edits. Pseudorandomness plausibly holds against subexponential-time distinguishers.*

For the above, we need the permuted codes conjecture to hold specifically for *Reed-Solomon codes*. Note that while the Reed-Solomon code has a large alphabet, the resulting PRCs are over a binary alphabet. By using a standard transformation from PRCs to watermarking schemes [CG24], we obtain an LLM watermark with the analogous guarantees:

Theorem (An improved watermark, Theorem 5.14)**.** *Under (a slight variant of) the permuted codes conjecture, for any constant $\alpha > 0$, there exists an undetectable, $\widetilde{\Omega}(\alpha^7)$-edit-robust watermark requiring per-token entropy of only α.*

For this result, we need a slight variant of the permuted codes conjecture to hold for *folded Reed-Solomon Codes*, with the caveat that these are technically not linear codes over their underlying alphabet.

This is the first undetectable LLM watermark that is robust to a constant rate of edits, and which works under a constant per-token entropy rate. Furthermore, our edits can be worst-case (i.e., the watermark is strongly adaptively robust). Previous watermarks either noticeably changed the distribution of text [KTHL24], or required a superconstant (and therefore unrealistic) rate of entropy [GM24].

1.2 Related work

Watermarks. The recent line of work on LLM watermarks was largely initiated by [Aar22, KGW+23]. Several works with provable guarantees followed [CGZ24, ZALW24, KTHL24]; however, until [CG24] no watermark was simultaneously undetectable and robust to significant modification. Subsequent works with both of these properties all use pseudorandom codes [GM24, GZS25, CHS25].

It is impossible to construct a watermark that is robust against arbitrary adversarial removal strategies, as a strong enough adversary can simply create an unwatermarked response itself. [ZEF+24] show a formal version of this impossibility, where they assume that the adversary can generate, from any given response, a uniformly random response of the same quality. As a result, we (and other watermarking works) consider robustness only to restricted adversarial removal strategies, e.g. those making a bounded number of edits to the LLM output.

We refer the reader to [ZGC+25] for further background on watermarks.

Quasipolynomial-time distinguishing attacks. The first pseudorandom code construction, of Christ and Gunn [CG24], is essentially a binary random linear code satisfying some planted $(\log n)$-sparse parity checks. Encryptions of '1' are random codewords with a constant rate of noise, which are pseudorandom under subexponential LPN. Encryptions of '0' are uniform strings. Decoding, which tolerates random substitutions, involves checking whether the given string satisfies a $\left(\frac{1}{2} + \frac{1}{\mathsf{poly}(n)}\right)$ fraction of the parity checks for a particular polynomial in n. This is true for encryptions of '1' if and only if the parity checks are sufficiently—in particular, $O(\log n)$—sparse. However, this sparsity also enables a quasipolynomial-time attack that brute-force guesses a parity check in $n^{O(\log n)}$ time.

Subsequent works follow a similar technique, embedding hidden $(\log n)$-sparse structure. For example, [GM24] constructs PRCs under the assumption that there exist $(\log n)$-local weak PRFs. [GG25] constructs PRCs under the assumption that a random hypergraph is indistinguishable from a random hypergraph with a planted structure consisting of $\log n$ hyperedges. These constructions exhibit the same phenomenon as [CG24]—$\log n$ sparsity is necessary for decoding, but it unfortunately leads to quasipolynomial-time distinguishing attacks that simply brute-force search for this sparse structure.

In fact, in Section 4.4 we show that a broader class of constructions which does not immediately appear to necessitate $(\log n)$-sized structures always has quasipolynomial-time distinguishing attacks. This class includes constructions where codewords are comprised in part of a random string r and a predicate $f(r)$ that tolerates a constant rate of errors.

We do not know how to construct PRCs with public encoding using our techniques, unlike the

constructions of [CG24, GG25].

Robustness notions. Robustness can vary on two important axes: the type and number of errors the channel is allowed to introduce, and the information the channel knows. The most basic notion of a PRC tolerates a constant rate of substitutions, and even constructing this object is challenging. However, ideally one wants to tolerate a constant rate of edits, allowing insertions and deletions in addition to substitutions. Orthogonal to the kind of errors is the knowledge the channel has when choosing these errors. A weak channel may make only random errors; a stronger channel may be computationally bounded, and choose errors without knowledge of the PRC secret key. The strongest kind of channel is computationally unbounded and receives the secret key as input—this is the error model we consider in this work. Note that while this channel is able to distinguish PRC codewords from random, this does not necessarily create an issue for robustness.

Two constructions go beyond substitution robustness. [CG24] propose a binary PRC that is robust to a constant rate of *random* deletions. [GM24] propose a polynomial-alphabet PRC that is robust to a constant rate of worst-case edits. However, this large alphabet translates to a watermark that requires unrealistically high-entropy responses. Accommodating realistic entropy rates requires a *binary* PRC with robustness to a constant rate of edits; this is what we construct in this work.

The only paper we are aware of that considers PRCs for channels which may depend on the secret key in some way is [AAC+25]. They defined a stronger notion of *ideal* security for PRCs, where the channel is computationally bounded but can adaptively query encoding and decoding oracles. They showed that the original PRC of [CG24] can be easily transformed into one with ideal security. In contrast, we present PRCs which are robust even against an error channel that is computationally unbounded and knows the secret key.

On the hardness of constructing PRCs. Two recent concurrent works [GGW25, DMR25] show that PRCs are difficult to construct in a formal sense. That is, they cannot be black-box constructed from a wide range of cryptographic primitives including random oracles, public-key encryption, and virtual black-box obfuscation. Therefore particular cryptographic assumptions—such as our permuted codes conjecture or the prior sub-exponentially secure LPN [CG24, GG25], planted hyperloop [GG25], or weak PRF [GM24] assumptions—are necessary. It remains an interesting open question to construct sub-exponentially secure pseudorandom codes from more standard assumptions than the permuted codes conjecture, such as LPN or LWE or variants thereof (sparse LPN, dense–sparse LPN, ring LWE).

"Heuristic construction" of [CG24]. In addition to their LPN-based PRC, [CG24] mentions an alternate construction, which is "heuristic" in that it lacks strong evidence for pseudorandomness. This construction involves taking a binary error-correcting code, permuting its symbols, and applying a constant rate of random substitutions. [CG24] does not specify the codes for which this construction should be pseudorandom, though they suggested investigating polar codes due to their high rate.

It is fairly immediate that pseudorandomness of their heuristic construction is equivalent to the permuted codes conjecture, when both are restricted to binary codes with polynomial dual distance. The only difference is that [CG24] omits the alphabet permutation. However, for binary codes the alphabet permutation is meaningless as it is simply a one-time pad, which can be removed by adding pairs of codewords.

Furthermore, our result that the alphabet permutation is necessary (Theorem 4.5) implies that

the heuristic construction is insecure when instantiated with larger-alphabet codes (though [CG24] considers it only for binary codes).

2 Preliminaries

Let $\mathrm{Bin}(n,p)$ denote the binomial distribution with n independent trials and success probability p of each trial. Let the *substitution channel* $\mathrm{SC}_\eta : \Sigma^\star \to \Sigma^\star$ be the randomized channel which, on input $x \in \Sigma^\star$, outputs a string $y \in \Sigma^\star$ of the same length as x obtained by independently replacing each symbol of x with a uniformly random symbol from Σ with probability η. For a set X, we let S_X denote the symmetric group on X, i.e., the set of permutations $\pi : X \to X$. We say that a non-negative valued function $f(\lambda)$ of a security parameter λ is *negligible* (and write $f(\lambda) \leq \mathsf{negl}(\lambda)$) if f decays faster than any polynomial in λ, i.e., $f(\lambda) \leq O(1/\operatorname{poly}(\lambda))$ for any polynomial poly. While we state our guarantees with this "sub-polynomial" notion of decay, we emphasize that our constructions achieve *sub-exponential* security assuming sub-exponential security of the permuted codes conjecture (and this follows immediately from our proofs).

For a finite set S, we write $x_1, \ldots, x_k \leftarrow S$ to mean elements $x_1 \ldots, x_k$ sampled uniformly at random from S, with replacement. For a distribution D, we write $x_1, \ldots, x_k \leftarrow D$ to denote sampling k i.i.d. elements from D.

We will often use $\Sigma = \mathbb{F}_q$, the finite field of size q. In cases when the algebraic structure is not important, we will sometimes refer to elements of $\mathbb{F}_q$ as elements of $[q]$.

2.1 Pseudorandom codes

Definition 2.1. Let Σ be an alphabet and $\mathcal{E} : \{0,1\}^* \times \Sigma^\star \to \Sigma^\star$ be an error channel on $\Sigma^\star$ that may depend on some auxiliary information $\mathsf{sk} \in \{0,1\}^*$. A *(secret-key) pseudorandom code with strong adaptive robustness to* $\mathcal{E}$ is a tuple of polynomial-time randomized algorithms $(\mathsf{KeyGen}, \mathsf{Encode}, \mathsf{Decode})$, where:

- (Syntax) For a security parameter λ and functions $s(\lambda), n(\lambda)$ denoting the length of the secret key and the block length, we have: $\mathsf{KeyGen} : \{1^\lambda\} \to \{0,1\}^{s(\lambda)}$, $\mathsf{Encode} : \{0,1\}^{s(\lambda)} \to \Sigma^{n(\lambda)}$, and $\mathsf{Decode} : \{0,1\}^{s(\lambda)} \times \Sigma^{n(\lambda)} \to \{\mathsf{True}, \mathsf{False}\}$.
- (*Undetectability*; also referred to as *Pseudorandonmess*) For any polynomial-time distinguisher Dist, it holds that
$$\left| \Pr_{\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda)} \left[\mathsf{Dist}^{\mathsf{Encode}(\mathsf{sk})}(1^\lambda) = 1 \right] - \Pr_{\mathcal{U}} \left[\mathsf{Dist}^{\mathcal{U}}(1^\lambda) = 1 \right] \right| \leq \mathsf{negl}(\lambda),$$
where $\mathcal{U}$ denotes the uniform oracle which outputs a uniformly random string in $\Sigma^{n(\lambda)}$ each time it is called.
- (*Soundness*) For any fixed $y \in \Sigma^\star$, it holds that
$$\Pr_{\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda)} (\mathsf{Decode}(\mathsf{sk}, y) = \mathsf{True}) \leq \mathsf{negl}(\lambda).$$
- (Strongly-adaptive robustness) For any $\lambda \in \mathbb{N}$, we have that
$$\Pr_{\substack{\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda) \\ x \leftarrow \mathsf{Encode}(\mathsf{sk})}} (\mathsf{Decode}(\mathsf{sk}, \mathcal{E}(\mathsf{sk}, x)) = \mathsf{False}) \leq \mathsf{negl}(\lambda).$$

2.2 Watermarking schemes

Recall that one of our main motivations behind pseudorandom codes is to obtain *watermarking schemes* for autoregressive language models. An *autoregressive language model* Model over some alphabet Σ is a algorithm Model which takes as input a prompt $\mathtt{PROMPT} \in \Sigma^\star$ and a sequence of previous tokens $\mathsf{t}_1, \ldots, \mathsf{t}_{i-1}$ and outputs a distribution $\mathsf{Model}(\mathtt{PROMPT}, \mathsf{t}_{1:i-1}) \in \Delta(\Sigma)$ over the next token $\mathsf{t}_i \in \Sigma$. Given an integer $\ell \in \mathbb{N}$, such an algorithm defines a distribution over sequences of tokens $\mathsf{t} \in \Sigma^\star$, by repeatedly drawing $\mathsf{t}_i \sim \mathsf{Model}(\mathtt{PROMPT}, \mathsf{t}_{1:i-1})$ for $i \in [\ell]$. A *watermarking scheme* $\mathcal{W}$ for a language model Model consists of the following components (see Section 5.5 for a formal treatment):

- $\mathsf{Setup}(\lambda)$, which takes as input a security parameter λ and outputs a secret key sk of length polynomial in λ;
- $\mathsf{Wat}(\mathsf{sk}, \mathtt{PROMPT})$, which takes as input a secret key sk and a prompt $\mathtt{PROMPT}$ and outputs a sequence $\mathsf{t} \in \Sigma^\ell$;
- $\mathsf{Detect}(\mathsf{sk}, \mathsf{t})$, which takes as input a secret key sk and a sequence $\mathsf{t} \in \Sigma^\ell$ and outputs a response in $\{\mathsf{True}, \mathsf{False}\}$ indicating whether Detect detects the sequence t as being watermarked according to sk.

Generally speaking, we want the watermarking scheme $\mathcal{W}$ to have properties paralleling that of pseudorandom codes: in particular, we desire (a) *soundness*, meaning that any fixed string is detected as watermarked with negligible probability (Definition 5.12), (b) *undetectability*, meaning that strings output by $\mathsf{Wat}(\mathsf{sk}, \mathtt{PROMPT})$ are computationally indistinguishable from strings output by repeatedly sampling from Model (Definition 5.11), and (c) *strong adaptive robustness*, meaning that any channel which corrupts the watermarked string by a small number of edits, *even with knowledge of the secret key*, cannot fool the detection algorithm Detect (Definition 5.13).

In this work we omit discussion of multi-bit watermarks, although our results immediately yield watermarks with positive information rate. See [CG24, Sections 2.6 and 7.4] for the applications of PRCs encoding multiple bits to watermarking with unforgeable public attribution.[4]

3 Technical overview

We now overview the technical ideas behind our main contributions. In Section 3.1, we describe the permuted codes assumption. We explain how it relates to the "permuted puzzles" assumption of prior work, we show that it holds *statistically* for a small number of samples if the alphabet is small, and we give an attack on large-alphabet schemes that do not use the alphabet permutation. We then reiterate the need for new approaches to PRC constructions by showing a quasipolynomial-time attack against a general blueprint used in prior works. In Sections 3.2 and 3.3, we outline how this assumption yields binary pseudorandom codes with strongly-adaptive robustness to edits and sub-exponential security.

[4]Some works have referred to similar properties as "public detection" or "publicly detectable watermarks." In [CG24, Sections 2.6 and 7.4] it is explained why unforgeable attribution must be treated as an independent property from standard watermark detection.

3.1 The computational assumption: permuted codes

We introduce a family of assumptions which we collectively call *permuted codes* assumptions. Roughly speaking, these assumptions posit that, for codes C satisfying certain properties, taking a random codeword from C, randomly permuting it and adding noise is indistinguishable from simply outputting a uniformly random string. To formally state the assumption, we fix a set Σ denoting the alphabet, an integer $n \in \mathbb{N}$ denoting the block length, and let $C \subseteq \Sigma^n$ denote an arbitrary set (which will typically be taken to be an error-correcting code with large dual distance). Given a real number $\eta > 0$, we let the *substitution channel* $\mathrm{SC}_\eta : \Sigma^\star \to \Sigma^\star$ be the randomized channel which, on input $x \in \Sigma^\star$, outputs a string $y \in \Sigma^\star$ obtained by independently replacing each symbol of x with a uniformly random symbol from Σ with probability η. That is,

$$\mathrm{SC}_\eta(x)_i = \begin{cases} x_i & \text{with probability } 1-\eta, \text{ and} \\ \text{uniform in } \Sigma & \text{with probability } \eta. \end{cases}$$

We define $\mathcal{D}_{n,\Sigma,C,\eta,T}$ as follows:

1. Sample random alphabet permutations $\pi_1, \ldots, \pi_n \leftarrow S_\Sigma$ and index permutation $\sigma \leftarrow S_{[n]}$.
2. Repeat T times:
   (a) Sample $c \leftarrow C$.
   (b) Define $\hat{c}$ by $\hat{c}_i \leftarrow \pi_i(c_{\sigma(i)})$ for $i \in [n]$.
   (c) Output $\mathrm{SC}_\eta(\hat{c})$.

The permuted codes assumption says that $\mathcal{D}_{n,\Sigma,C,\eta,T}$ is pseudorandom.

Definition 3.1 (Permuted codes assumption)**.** Let $\lambda \in \mathbb{Z}^+$ be a security parameter, and let $n = n(\lambda)$, $q = q(\lambda)$, $T = T(\lambda)$ be polynomially-bounded functions in λ denoting the block length, alphabet size, and number of samples, respectively. Let $C = C(\lambda) \subseteq \mathbb{F}_q^n$ be a family of codes. The *permuted codes assumption for C with error η* states that $\mathcal{D}_{n,\mathbb{F}_q,C,\eta,T}$ is computationally indistinguishable from $\mathrm{Unif}((\mathbb{F}_q^n)^T)$.

Note that the uniform distribution $\mathrm{Unif}((\mathbb{F}_q^n)^T)$ is identical to the distribution $\mathcal{D}_{n,\mathbb{F}_q,\mathbb{F}_q^n,\eta,T}$ where $C = \mathbb{F}_q^n$ is the trivial code. We conjecture that the permuted codes assumption holds with any constant error rate for any family of linear codes with polynomial dual distance.

Conjecture 3.1 (General permuted codes conjecture)**.** *Let $\lambda \in \mathbb{Z}^+$ be a security parameter, and let $n = n(\lambda)$, $q = q(\lambda)$, $T = T(\lambda)$ be polynomially-bounded functions in λ denoting the block length, alphabet size, and number of samples, respectively. The* permuted codes conjecture *states that if $C = C(\lambda) \subseteq \mathbb{F}_q^n$ is any family of linear codes with dual distance $d = d(\lambda) = \lambda^{\Omega(1)}$ and $\eta = \Omega(1)$ is any constant error rate, then $\mathcal{D}_{n,\mathbb{F}_q,C,\eta,T}$ is computationally indistinguishable from* $\mathrm{Unif}((\mathbb{F}_q^n)^T)$.

In Section 6, we show that this conjecture immediately yields a binary PRC that is robust to a constant rate of *substitutions*. That is, let C be an efficiently decodable binary code that has high dual distance. Algorithm Encode of our PRC simply outputs permuted random noisy codewords, which are pseudorandom under the permuted codes assumption. Decode inverts the permutation and applies the efficient decoder. A far more significant challenge, which we discuss in the next section (Section 3.2) is achieving *edit* robustness.

For our results, we will only need the permuted codes assumption for particular families of codes; below we state the specialization to the important case where C is the family of *Reed-Solomon codes* (Definition 5.1).

Conjecture 3.2 (Permuted Reed-Solomon conjecture)**.** *Fix any constant η. For security parameter λ, let C be the Reed-Solomon code $\mathsf{RS}_{\mathbb{F}_q,n,k}$ with $n = \lambda, q = \lambda - 1, k = \lambda^{1/5}$ (see Definition 5.1).*[5] *Then the distributions $\mathcal{D}_{n,\mathbb{F}_q,C,\eta,T}$ and $\mathrm{Unif}((\mathbb{F}_q^n)^T)$ are computationally indistinguishable for any $T = \mathrm{poly}(\lambda)$.*

We also specialize Definition 3.1 to folded Reed-Solomon codes (Conjecture 5.3), which we use to build edit-robust watermarks in Section 5.

Evidence: as a consequence of permuted puzzles Definition 3.1 (and its specialization Conjecture 3.2) are related to various precedents in the literature. The closest connection is to the *permuted puzzles* assumption of [BHMW21, BW21], which is similar to Definition 3.1 but applies a single permutation to all of $[n] \times \Sigma$. To formally state it, given $n \in \mathbb{N}$, $C \subseteq \Sigma^n$, and $T, m \in \mathbb{N}$, we define the distribution $\mathcal{D}^{\mathsf{PP}}_{n,\Sigma,C,T,m}$ as follows:

1. Sample a random permutation $\pi \leftarrow S_{[n]\times\Sigma}$.
2. Repeat T times:
   (a) Sample $c \leftarrow C$.
   (b) Sample $i_1, \ldots, i_m \leftarrow [n]$.
   (c) Output $(\pi(i_1, c_{i_1}), \ldots, \pi(i_m, c_{i_m}))$.

Conjecture 3.3 (Permuted puzzles, "General conjecture" [BHMW21, BW21])**.** *Let $\lambda \in \mathbb{Z}^+$ denote a security parameter, and let $n = n(\lambda), q = q(\lambda), T = T(\lambda), m = m(\lambda) \le n(\lambda)$ be polynomially growing functions in λ denoting the block length, alphabet size, number of samples, and number of symbols per codeword, respectively. If $C = C(\lambda) \subseteq \mathbb{F}_q^n$ is any family of linear codes with dual distance $d = d(\lambda) = \lambda^{\Omega(1)}$ and $m \le (1 - \Omega(1)) \cdot n$, then $\mathcal{D}^{\mathsf{PP}}_{n,\mathbb{F}_q,C,T,m}$ and $\mathcal{D}^{\mathsf{PP}}_{n,\mathbb{F}_q,\mathbb{F}_q^n,T,m}$ are computationally indistinguishable.*

Conjecture 3.3 was originally introduced for the unrelated purpose of obtaining doubly-efficient private information retrieval. Interestingly, we can show that the permuted codes conjecture (Conjecture 3.1) *is implied by* the permuted puzzles conjecture (Conjecture 3.3), meaning that we can base the existence of PRCs on either. In fact, the permuted puzzles conjecture is equivalent to the version of the permuted codes conjecture where the substitution channel is replaced by the erasure channel.

Proposition 3.4. *Suppose that the permuted puzzles conjecture (Conjecture 3.3) holds. Then the permuted codes conjecture (Conjecture 3.1) holds.*

The proof of Proposition 3.4 proceeds by drawing enough samples from either $\mathcal{D}^{\mathsf{PP}}_{n,\mathbb{F}_q,C,T,m}$ or $\mathcal{D}^{\mathsf{PP}}_{n,\mathbb{F}_q,\mathbb{F}_q^n,T,m}$ to learn which elements $z \in [n]\times\Sigma$ never appear in the same sample $(\pi(i_1, c_{i_1}), \ldots, \pi(i_m, c_{i_m}))$. This allows us to convert samples from $\mathcal{D}^{\mathsf{PP}}_{n,\mathbb{F}_q,C,T,m}$ or $\mathcal{D}^{\mathsf{PP}}_{n,\mathbb{F}_q,\mathbb{F}_q^n,T,m}$ into samples from $\mathcal{D}_{n,\mathbb{F}_q,C,\eta,T}$ or $\mathrm{Unif}((\mathbb{F}_q^n)^T)$ respectively. See Section 4 for the full proof.

[5] The choice of $k = \lambda^{1/5}$ is unimportant; it is straightforward to adjust our algorithm to accommodate any k which grows polynomially with λ.

Evidence: statistical uniformity of a few samples. Next, we discuss a complementary piece of evidence for Definition 3.1, which results from asking the following question: *If we take $T(\lambda)$ to be very small (e.g., $O(\log n)$), can we prove that $\mathcal{D}_{n,\Sigma,C,\eta,T}$ and $\mathrm{Unif}((\Sigma^n)^T)$ are in fact statistically indistinguishable?* When the size of the alphabet Σ is large (e.g., a sufficiently large polynomial in n), it is easy to see that this is not the case, by a simple counting argument. It is also straightforward to see that such statistical indistinguishability does not hold when $T = \omega(\log n)$, simply because there is not enough entropy in the sampling process of $\mathcal{D}_{n,\Sigma,C,\eta,T}$. However, when $|\Sigma| = O(1)$, there exists $T(\lambda) = \Omega(\log d)$ such that the two distributions in question are statistically close:

Proposition 3.5 (Informal version of $q = O(1)$ case of Corollary 4.3)**.** *Let $C \subseteq \mathbb{F}_q^n$ be any code with dual distance d and constant alphabet size q. Fix a constant $\eta > 0$. For sufficiently small constant c, any $T \leq c \cdot \log d$ satisfies*

$$d_{\mathsf{TV}}(\mathcal{D}_{n,\mathbb{F}_q,C,\eta,T}, \mathrm{Unif}((\mathbb{F}_q^n)^T)) \leq \exp(-\Omega(d)).$$

The proof of Proposition 3.5 proceeds by noting that, due to the permutation $\sigma : [n] \to [n]$ of the n positions, for any distinguisher Dist between samples from either $\mathcal{D}_{n,\mathbb{F}_q,C,\eta,T}$ or $\mathcal{D}_{n,\mathbb{F}_q,\mathbb{F}_q^n,\eta,T}$ we can write Dist in the following form. For each position $i \in [n]$, we may group the bits at position i across the T samples, and interpret the group as an element of $\mathbb{F}_q^T$; the distinguisher Dist, in turn, must depend only on the multinomial counts of the n elements of $\mathbb{F}_q^T$ corresponding to a codeword from either $\mathcal{D}_{n,\mathbb{F}_q,C,\eta,T}$ or $(\mathbb{F}_q^n)^T$.

In turn, *arbitrary* algorithms which depend only on these counts turn out to be very simple: they can be written as length-nT, width-$\binom{n+q^T-1}{q^T-1}$ read-once branching programs, which follows from the fact that there are at most $\binom{n+q^T-1}{q^T-1}$ possibilities for what the collection of counts of the n elements of $\mathbb{F}_q^T$ can be. By considering a special form of the Fourier decomposition of such branching programs (Lemma 4.1) due to [FK18], we can show that the output of such programs are close under the distributions $\mathcal{D}_{n,\mathbb{F}_q,C,\eta,T}$ and $(\mathrm{Unif}(\mathbb{F}_q^n)^T)$. Very roughly, the reason is that the low-degree Fourier terms are identical due to the high dual distance of C, while high-degree Fourier terms are negligible due to the noise introduced by SC_η. We note that our argument easily implies pseudorandomness against *linear tests*, a benchmark sometimes used to judge new LPN-style cryptographic conjectures [CRR21, BCG+22, DJK25]. See Section 4.2 for the proof.

While the setting of Proposition 3.5 only applies in the special case where $|\Sigma|$ is relatively small, as we observe in Section 6, the permuted codes assumption (Definition 3.1) with these parameter settings implies strongly-adaptive robust pseudorandom codes (with subexponential security, assuming such of Definition 3.1) against *substitutions*, which is not known to be achievable under any other assumptions. To handle *edits*, we need to rely on the permuted codes assumption for (folded) Reed-Solomon codes with a large alphabet, which are not covered by the above result.

Cryptanalysis: the alphabet permutation is necessary. The permuted codes assumption states that a distribution of "scrambled" codewords is pseudorandom. Recall that this scrambling involves several components: First, the codeword indices are permuted by a index permutation $\sigma \in S_{[n]}$. Second, an *alphabet permutation* $\pi_i \in S_\Sigma$ is applied to each codeword symbol. Finally, random substitution errors are applied to this doubly permuted codeword.

A natural question is whether all of these steps are necessary for pseudorandomness. Prior work on the permuted puzzles conjecture gave a partial answer—a "toy conjecture" put forth in [BIPW17] asserted that the scrambled codewords were pseudorandom *even without* the random

substitution errors. However, this was shown to be false in [BHMW21, BW21], which suggested amending the conjecture to apply random erasures—this is exactly Conjecture 3.3. The analogous question could be asked about the permutations. With *no* permutations, any efficiently decodable code would break the conjecture, with the decoding algorithm serving as a distinguisher. But could the index permutation alone be enough?

We show that the alphabet permutations are necessary for the permuted codes assumption in Theorem 4.5. That is, there exists a family of codes with polynomial dual distance that can be efficiently distinguished from random when codewords are position-permuted and subjected to a constant rate of random substitutions. This code family is simply Reed-Solomon codes.

Suppose we are given m position-permuted Reed-Solomon codewords $c^{(1)}, \ldots, c^{(m)}$ with a constant rate of substitutions. Our attack attempts to find a low-degree nonzero multivariate polynomial f such that $f(c_i^{(1)}, \ldots, c_i^{(m)}) = 0$ for all $i \in [n]$. If m is small enough, there will be a significant number of i's with no noise across all codewords; these corresponding evaluation points take the form $(p_1(\alpha_i), \ldots, p_m(\alpha_i))$ for low-degree polynomials $p_1, \ldots, p_m$ and some unknown α_i. These points can be annihilated by a far lower-degree polynomial than one would expect of random points. Therefore, for a proper choice of degree of f, this interpolation task is possible for the scrambled codewords but not for random points.

This attack can also be viewed as an attack against the McEliece cryptosystem instantiated with Reed-Solomon codes. This attack uses only ciphertexts, and does not require the public key.

Cryptanalysis: a quasipolynomial-time attack against a broad class of constructions. To reiterate the need for new approaches to PRCs, we show that all constructions following a natural blueprint are vulnerable to quasipolynomial-time attacks. In such constructions, codewords are binary strings that include a random string r and a noise-tolerant predicate $f(r)$. That is, codewords are of the form $\sigma(r||f(r)||y)$ where $\sigma \in S_{[n]}$ is fixed across all codewords, $r \leftarrow \{0,1\}^\ell$, and y is drawn from any distribution over $\{0,1\}^{n-\ell-1}$.[6] f satisfies $\Pr_{r,r' \leftarrow \mathrm{SC}_\eta(r)}[f(r) = f(r')] \geq 1/2 + 1/\operatorname{poly}(n)$, for some constant $\eta > 0$.

In other words, codewords have a planted relation between r and $f(r)$, which is hidden by the permutation. Since f is noise-tolerant, this relation still holds with significant probability when noise is added, yielding a weak decoder (that can be boosted to a full decoder). Several existing constructions fall into this class [CG24, GM24].

Our attack is fairly simple: Since f is noise-tolerant, it must have a Fourier coefficient at degree $t = O(\log n)$ with weight $n^{-O(\log n)}$. Therefore, there is a quasipolynomial-time distinguisher that brute-force estimates all Fourier coefficients up to degree t. We prove this in Section 4.4.

Note that our new binary constructions do not fall into this class. Instead, our codewords can be viewed as containing a random string $r \in \{0,1\}^\ell$, where ℓ is the dual distance of the code, and a *longer-output function* $F : \{0,1\}^\ell \to \{0,1\}^{n-\ell}$. No single bit of F is noise-tolerant, and decoding uses the entire output of F. Furthermore, polynomial dual distance of the codes used in our constructions prevents any similar attack.

3.2 Edit-robust pseudorandom codes from permuted codes assumption

In this section we explain how the permuted Reed-Solomon conjecture (Conjecture 3.2), which is a special case of the permuted codes assumption (Definition 3.1), yields an edit-robust pseudorandom

[6] Abusing notation a bit, we mean that the ith bit of the codeword is the $\sigma(i)^{\text{th}}$ bit of $(x||f(x)||y)$.

code.

Fix a prime power q denoting the alphabet size (so that $\Sigma = \mathbb{F}_q$), and let $n = q - 1$; we will consider the Reed-Solomon code $\mathsf{RS}_{\mathbb{F}_q,n,k}$ with $k = n^{1/5}$.[7] We construct a pseudorandom code PRC as follows (see Algorithm 1 for the full description).

Key Generation. The secret key sk consists of the permutations $\sigma \leftarrow S_{[n]}, \pi_1, \ldots, \pi_n \leftarrow S_{\mathbb{F}_q}$ as in the definition of $\mathcal{D}_{n,\mathbb{F}_q,C,\eta,T}$ (see Section 3.1).

Encoding. To produce a PRC codeword, we first generate one sample (corresponding to one index $t \in [T]$) from the distribution $\mathcal{D}_{n,\mathbb{F}_q,C,\eta,T}$, which may be expressed as a vector $z \in \mathbb{F}_q^n$. Note that the data of z can be be equivalently expressed as a list of tuples $(1, z_1), \ldots, (n, z_n)$. Roughly speaking, to output a PRC codeword, we sample m indices $i_j \sim \mathrm{Unif}([n])$ and write the corresponding tuples (i_j, z_{i_j}) in binary, yielding a string in $\{0,1\}^{\log(nq)}$ for $j \in [m]$, and output their concatenation. (Technically, we need to take some additional care because if i_j is sampled twice it will be followed by (binary) z_{i_j} both times, which is unlikely for a random string. The necessary modifications are straightforward.)

Decoding. The technical bulk of the algorithm (and the proof that it yields an edit-robust PRC) lies in the decoding algorithm. A key ingredient is the *list recoverability* of Reed-Solomon codes, which generalizes the somewhat better-known notion of *list decoding*. In particular, let us consider the Reed-Solomon code $C = \mathsf{RS}_{\mathbb{F}_q,n,k}$ and fix parameters $\ell \in \mathbb{N}, \zeta \in [0,1]$ satisfying $\zeta n \geq \sqrt{k\ell n}$. Then the code C is efficiently (ℓ, ζ)-*list recoverable*, which means that there is a $\mathrm{poly}(n)$-time algorithm which takes as input n sets $\mathcal{L}_1, \ldots, \mathcal{L}_n \subset \mathbb{F}_q$ satisfying $|\mathcal{L}_i| \leq \ell$ for each $i \in [n]$, and outputs a list consisting of all codewords $c' \in C$ satisfying $c'_i \in \mathcal{L}_i$ for at least ζn values of $i \in [n]$. (In particular, the number of such codewords is at most $\mathrm{poly}(n)$; see Theorem 5.1 for a formal statement.)

Recall from the encoding algorithm above and the definition of $\mathcal{D}_{n,\mathbb{F}_q,C,\eta,T}$ (Definition 3.1) that a PRC codeword y consists of the binary encoding of a collection of pairs $(i_j, \pi_{i_j}(c_{\sigma(i_j)}) + e_{i_j}) \in [n] \times \mathbb{F}_q$ where $i_j \in [n]$ is an index, $\pi_{i_j}(c_{\sigma(i_j)})$ is a permuted symbol from a noisy Reed-Solomon codeword c, and e_{i_j} is a random substitution error (which in particular is 0 with probability $1 - \eta$). The crucial insight is as follows: for any channel $\mathcal{E}_{\mathsf{adv}}$ which corrupts its input y by at most $\varepsilon_{\mathsf{E}} \cdot |y|$ edits (for a sufficiently small constant ε_{E}) to some value y', many pairs $(i_j, \pi_{i_j}(c_{\sigma(i_j)}) + e_{i_j})$ must remain "relatively intact" somewhere inside y'. That is, many pairs will contain at most $O(\log(nq) \cdot \varepsilon_{\mathsf{E}})$ edits. Moreover, for any string $w \in \{0,1\}^{\log(nq)}$, the number of strings w' of edit distance at most $O(\log(nq) \cdot \varepsilon_{\mathsf{E}})$ from w is at most $(nq)^{O(\varepsilon_{\mathsf{E}} \log(1/\varepsilon_{\mathsf{E}}))}$. Thus, given an input y' to the decoding algorithm, we can "brute force" over all strings close in edit distance to each $O(\log(nq))$-length contiguous substring of y, and construct corresponding lists as input to a list recovery algorithm for C. Under an appropriate choice of parameters, the list recovery algorithm will output a nonempty set if in fact $y' = \mathcal{E}_{\mathsf{adv}}(y)$ for some output y of the PRC encoding algorithm.

In more detail, the decoding algorithm acts as follows, given some input string y'. We first initialize lists $\mathcal{L}_1, \ldots, \mathcal{L}_n \leftarrow \emptyset$ (which will ultimately be the input to a list recovery procedure for C). We inspect all contiguous substrings of length $O(\log(nq))$ inside y', and for each, compute all strings within edit distance $O(\log(nq) \cdot \varepsilon_{\mathsf{E}})$. In turn, for each of these strings, we interpret them as the binary encoding of a pair $(i, \pi_i(a))$ for some $a \in \mathbb{F}_q$ and $i \in [n]$. We then add $\pi_i^{-1}(a)$ to the

[7] Any k which grows polynomially with n suffices for our applications.

list $\mathcal{L}_{\sigma(i)}$ (using our knowledge of π_i, σ as encoded by the secret key sk). Finally, we run the list recovery algorithm for C on input $\mathcal{L}_1, \ldots, \mathcal{L}_n$. If the output list contains any codeword $c' \in C$, we output True; otherwise, we output False. See Algorithm 1 for the full description.

Desired properties of the PRC. The robustness of the decoding algorithm follows from the above discussion: if $y' = \mathcal{E}_{\mathsf{adv}}(y)$ for some PRC codeword y, then for many indices $i \in [n]$, the list $\mathcal{L}_i$ will contain the correct symbol c^*_i from the Reed-Solomon codeword c^* corresponding to y (since we will have "guessed" it via the appropriate search over all $O(\log(nq) \cdot \varepsilon_{\mathsf{E}})$-edit close strings). Moreover, using our bound of $(nq)^{O(\varepsilon_{\mathsf{E}} \log(1/\varepsilon_{\mathsf{E}}))}$ on the size of an $O(\log(nq) \cdot \varepsilon_{\mathsf{E}})$-edit distance ball around any string in $\{0,1\}^{\log(nq)}$, we may bound the size of the lists $\mathcal{L}_i$ by some parameter ℓ satisfying $\sqrt{k\ell n} = o(n)$. Therefore, by the list recoverability of Reed-Solomon codes, the list recovery algorithm will output c^*. See Lemmas 5.9 and 5.11 for the formal arguments.

Moreover, soundness of the PRC follows from a simple counting argument which, roughly speaking, computes the number of strings y'' which have enough contiguous substrings of length $O(\log(nq))$ which are $O(\log(nq) \cdot \varepsilon_{\mathsf{E}})$-edit close to some contiguous substring of the input y' to the decoding algorithm; see Lemma 5.12.

Finally, the undetectability of the PRC construction follows directly from Conjecture 3.2, since the encoding algorithm above may be interpreted as outputting a (randomized) function F of one sample from $\mathcal{D}_{n,\mathbb{F}_q,C,\eta,T}$. Moreover, if we instead pass one sample from $\mathcal{D}_{n,\mathbb{F}_q,\mathbb{F}_q^n,\eta,T}$ to this randomized function F, the output distribution may be verified to be the uniform distribution. Thus, any algorithm which distinguishes between PRC codewords and uniform strings may be used to distinguish between samples from $\mathcal{D}_{n,\mathbb{F}_q,C,\eta,T}$ and $\mathrm{Unif}((\mathbb{F}_q^n)^T)$; see Lemma 5.13.

3.3 Edit-robust watermarking schemes using folded Reed-Solomon codes

Next, we discuss how the ideas from the previous section can be extended to obtain edit-robust watermarking schemes for autoregressive language models. As we will see, this requires PRCs that are robust to a very large fraction of edits, close to $\frac{1}{2}$. Above we constructed PRCs that are only robust to some small constant fraction of edits. However, we'll see we can extend the ideas to handle a large fraction of edits by relying on folded Reed-Solomon codes.

We begin by describing a generic transformation that converts any pseudorandom code into a watermarking scheme, due to [CG24]. In particular, consider a language model Model (see Section 2.2) and suppose we are given a pseudorandom code $\mathsf{PRC} = (\mathsf{KeyGen}, \mathsf{Encode}, \mathsf{Decode})$ with some block length n. For simplicity, we assume that the alphabet Σ for both Model and PRC is $\Sigma = \{0,1\}$, which is essentially without loss of generality (see Section 7.1 of [CG24]). The $\mathsf{Setup}(1^\lambda)$ function of the watermarking scheme simply calls the KeyGen procedure for PRC, which outputs some sk. To implement the watermarking function $\mathsf{Wat}(\mathsf{sk}, \texttt{PROMPT})$, we first generate a PRC codeword $x \leftarrow \mathsf{Encode}(\mathsf{sk})$, $x \in \Sigma^n$, and then generate a sequence of tokens $\mathsf{t}_{1:n}$ by attempting to generate each bit t_i so as to maximize its probability of being equal to x_i: in particular, if we have that $\mathsf{Model}(\mathsf{t}_i = 1 \mid \texttt{PROMPT}, \mathsf{t}_{1:i-1}) = p_i$, then we will draw t_i from the distribution $\mathrm{Ber}(p_i - (-1)^{x_i} \cdot \min\{p_i, 1-p_i\})$. This procedure is then repeated for multiple blocks of length n to reach some desired output length ℓ (see Algorithm 2); for simplicity, we assume that $\ell = n$ in this discussion. It is straightforward to see that the resulting distribution of $\mathsf{t}_{1:n}$ is identical to that produced by Model if x is in fact uniformly random; thus, if x is computationally indistinguishable from uniform, the output of Wat is computationally indistinguishable from that of Model (i.e., we have undetectability).

Finally, the $\mathsf{Detect}(\mathsf{sk}, \mathsf{t})$ function of the watermarking scheme simply calls the $\mathsf{Decode}(\mathsf{sk}, \mathsf{t})$ function for PRC, and outputs the same response. We begin by reviewing the argument in [CG24] which shows that the resulting watermarking scheme is robust to a constant fraction of *substitutions* (assuming that PRC has such substitution-robustness). To do so, we need to assume that *entropy* of the output of Model is sufficiently high, i.e., that each token has at least entropy α, on average. If this is the case, then the Hamming distance between the PRC codeword x and the output t of Wat is bounded above by $\left(\frac{1}{2} - \Omega(\alpha^2)\right) \cdot n$ (see Lemma 21 of [CG24], reproduced as Lemma 5.18). Now consider any substitution channel $\mathcal{E}_{\mathsf{adv}}$ which corrupts at most a fraction $c_0 \cdot \alpha^2$ of its input, where c_0 denotes a sufficiently small universal constant, and let $\mathsf{t}' = \mathcal{E}_{\mathsf{adv}}(\mathsf{t})$. Then the Hamming distance between x and t' is bounded above by $\left(\frac{1}{2} - \Omega(\alpha^2)\right) \cdot n$. Thus, if PRC is robust to any *substitution channel* which corrupts at most a fraction $\left(\frac{1}{2} - \Omega(\alpha^2)\right)$ of its input, then the watermarking scheme is robust to $\mathcal{E}_{\mathsf{adv}}$.

Our contribution: edit robustness via folded Reed-Solomon codes Can we extend the above argument to handle the case where $\mathcal{E}_{\mathsf{adv}}$ may introduce a constant fraction of *edits* (under the assumption that PRC has such robustness to edits)? While the above template naturally extends to handle edits, we need to address the following obstacle. The PRC watermarking scheme we described in Section 3.2 is robust to any channel which introduces a ε_{E}-fraction of edits, *for a sufficiently small constant* ε_{E}. However, in order to apply the above recipe for converting a PRC to a watermarking scheme, we need the PRC to be robust to $\left(\frac{1}{2} - \Omega(\alpha^2)\right)$-fraction of edits, for arbitrarily small constant α. This would be a significantly stronger result than what we showed above!

A closer inspection of the argument reveals that the issue is not merely with our analysis, i.e., using tighter bounds on the size of edit distance balls is insufficient to close this gap. In particular, for a given tuple $(i_j, \pi_{i_j}(c_{\sigma(i_j)})e_{i_j}) \in [n] \times [q]$ corresponding to one permuted symbol of a codeword $c \in \mathbb{F}_q^n$, the number of strings which could result after a fraction $\frac{1}{2} - \Omega(\alpha^2)$ of substitutions is at most $2^{\log(nq)\cdot(1-\Omega(\alpha^4))} = (nq)^{1-\Omega(\alpha^4)}$. Since, for the Reed-Solomon code, we must have $q \leq n$, this quantity approaches n^2 as $\alpha \to 0$, and in particular is $\omega(n)$. Aggregating over all $j \in [m] = \Theta(n)$, we see that the sum of the list sizes $\sum_{i=1}^{n} |\mathcal{L}_i|$ for the list recovery procedure must be $n \cdot (nq)^{1-\Omega(\alpha^4)} = n \cdot \omega(n) = \omega(n^2)$, which means the typical list $\mathcal{L}_i$ will be of size $\omega(n)$; this is too large for the list recovery guarantee for Reed-Solomon codes to apply (see Theorem 5.1).

The core issue here is that we are "wasting" bits encoding the indices i_j in the PRC encoding algorithm: if there were a way to avoid having to write them out explicitly, then the sum of the list sizes would be only $n \cdot n^{1-\Omega(\alpha^4)}$, which means the typical list size would be $n^{1-\Omega(\alpha^4)} = o(n)$, which would suffice for our needs. Unfortunately, our ability to write out these indices is crucial for edit robustness: if we had simply written the codeword symbols $c_{\sigma(i_j)}$ in order of e.g. increasing i_j, an adversary could introduce insertions or deletions to shift the position of these symbols by $\Theta(n)$ positions, and there is not a clear way to recover from this.

Instead, the main idea to get around this obstacle is encode multiple symbols in $\mathbb{F}_q$ using a single index i_j. Equivalently, we replace the Reed-Solomon code $\mathsf{RS}_{\mathbb{F}_q,n,k}$ with a *folded Reed-Solomon (FRS) code*, which is the code over alphabet $\mathbb{F}_q^m$ obtained by grouping together collections of m symbols from codewords of $\mathsf{RS}_{\mathbb{F}_q,n,k}$ (see Definition 5.2); here m should be interpreted as a constant, and will be taken to be $\mathrm{poly}(1/\alpha)$ in the context of the preceding discussion. As a FRS code is not linear over its alphabet[8] we cannot rely directly on the permuted codes assumption

[8] In particular, if we identify $\mathbb{F}_q^m \simeq \mathbb{F}_{q^m}$, then the folded Reed-Solomon code is not a linear code over $\mathbb{F}_{q^m}$. We

(Definition 3.1) to ensure undetectabilility of the resulting PRC. Nevertheless, the fact that the FRS code is linear over $\mathbb{F}_q$ (and has dual distance decreased by a factor of only $m = O(1)$ compared to that of the corresponding Reed-Solomon code) leads us to the natural generalization of Conjecture 3.2, in Conjecture 5.3 (in particular, Conjecture 5.3 is identical to Conjecture 3.2 except that the Reed-Solomon code is replaced with an FRS code with appropriate parameters).

Using a similar approach to our PRC construction with Reed-Solomon codes, we can show that a pseudorandom code based off of Conjecture 5.3 has robustness to any channel which introduces a $\left(\frac{1}{2} - \Omega(\alpha^2)\right)$-fraction of substitutions and a $c_1 \cdot \alpha^2$ fraction of edits, for some sufficiently small constant $c_1 > 0$. This is sufficient to obtain a watermarking scheme for language models with per-token entropy at least α: for a PRC codeword x used in the watermarking procedure Wat, at most a fraction $\left(\frac{1}{2} - \Omega(\alpha^2)\right)$ of substitutions are introduced in x to obtain the output t of Wat, and an adversary may introduce an additional $c_1 \cdot \alpha^2$ fraction of *edits* before the string is passed to the detection algorithm Detect. See Lemma 5.10 and Lemma 5.17 for the formal arguments.

4 The permuted codes assumption

In this section, we discuss the permuted codes assumption (Definition 3.1) and perform some cryptanalysis: first, in Section 4.1, we show that the permuted codes assumption is implied by the permuted puzzles assumption of [BHMW21, BW21] (Conjecture 3.3). In Section 4.2, we give some statistical evidence for Definition 3.1 when the alphabet size is constant by showing that logarithmically many permuted codewords are statistically uniform. Finally, in Section 4.3, we show that the permuted codes assumption *fails* if the alphabet permutation is dropped.

4.1 Comparison to permuted puzzles

We now show that the permuted puzzles assumption with $m = \Theta(n)$ implies the permuted codes assumption, for any constant $\eta > 1 - \frac{m}{n}$.

Proof of Proposition 3.4. Let C be as specified in Definition 3.1 and Conjecture 3.3, and let $m(\lambda) \geq \delta n(\lambda)$ for some constant $\delta \in (0, 1)$. Let η be a constant such that $\delta > 1 - \eta$.

We present an efficient algorithm $\mathcal{A}$ that takes in samples from $\mathcal{D}^{\mathsf{pp}}_{n,\mathbb{F}_q,C,T,m}$ or $\mathcal{D}^{\mathsf{pp}}_{n,\mathbb{F}_q,\mathbb{F}_q^n,T,m}$, and converts them to samples from $\mathcal{D}^{\mathsf{pc}}_{n,\mathbb{F}_q,C,\eta,T}$ or $\mathcal{D}^{\mathsf{pc}}_{n,\mathbb{F}_q,\mathbb{F}_q^n,\eta,T}$ respectively.

$\mathcal{A}$ is given $T \geq \lambda(nq)^2$ samples

$$\{(\alpha_1^{(i)}, \ldots, \alpha_m^{(i)})\}_{i \in [T]},$$

where each $\alpha_j^{(i)} \in [n] \times \mathbb{F}_q$. $\mathcal{A}$ initializes n empty sets $S_1, \ldots, S_n$.

For each S_ℓ, $\mathcal{A}$ does the following:

- For all $\alpha \in [n] \times \mathbb{F}_q$:
  - If α already appears in some set, continue.
  - If S_ℓ is empty, add α to S_ℓ.
  - If there exists $\alpha' \in S_\ell$ such that α and α' do not appear together in any of the T samples, add α to S_ℓ.

remark that the FRS code is linear over $\mathbb{F}_q$.

We'll argue that each S_ℓ is exactly $\{\alpha \in [n] \times \mathbb{F}_q : \exists j \in \mathbb{F}_q \text{ s.t. } \pi(i,j) = \alpha\}$ for some $i \in [n]$, where π is the random permutation in Conjecture 3.3.

We'll first show that all symbols added to S_ℓ correspond to the same index. Let S_ℓ contain some $\alpha \in [n] \times \mathbb{F}_q$, where $\alpha = \pi(i,j)$. Let $\alpha' = \pi(i',j')$ for $i' \neq i$. For a uniform $c \leftarrow C$, since C has dual distance at least 2, the marginal distribution $(c_{i'}, c_i)$ is uniform and $c_{i'} = j'$ and $c_i = j$ with probability $1/q^2$. The same holds for a uniform c. (Very coarsely), with probability at least $1/n^2$, i and i' are both included in the subsample in step (b). Therefore, for each sample, α and α' appear together with probability at least $1/(nq)^2$. The probability that α and α' do not appear together at any of the T samples is at most $(1 - 1/(nq)^2)^{\lambda(nq)^2}$, which is negligible.

On the other hand, if two symbols correspond to the same index, they will not appear together in any samples and they are indeed added to the same S_ℓ. Therefore, each S_ℓ has size exactly q.

$\mathcal{A}$ completes its conversion by drawing a random permutation $\pi_{\mathcal{A}} \leftarrow S_{[n]}$, and random bijections $\pi_1 : S_1 \to \mathbb{F}_q, \ldots, \pi_n : S_n \to \mathbb{F}_q$. For $\alpha \in [n] \times \mathbb{F}_q$, let $\phi(\alpha) := \pi_{\mathcal{A}}(\ell)$, where ℓ is the index of the set S_ℓ containing α.

$\mathcal{A}$ then constructs its ith sample for $i \in [T]$ as follows:

1. Initialize a random $\hat{c} \leftarrow \mathbb{F}_q^n$.
2. Sample $k \leftarrow \text{Bin}(n, 1-\eta)$. Let $I := \{\ell \in [n] : \exists j \in [n] \text{ s.t. } \ell = \phi(\alpha_j^{(i)})\}$. If $|I| < k$, output "fail."
3. Sample a random subset $I' \subseteq I$, where $|I'| = k$.
4. For all $j \in [m]$, let $\ell_j = \phi(\alpha_j^{(i)})$. If $\ell_j \in I'$, let $\hat{c}_{\ell_j} \leftarrow \pi_{\ell_j}(\alpha_j^{(i)})$.
5. Output $\hat{c}$.

The bulk of this work is in converting the m indices sampled without replacement in the permuted puzzles distribution, into the indices left error-free by the substitution channel in the permuted codes distribution. In each iteration, I is the set of (permuted according to $\pi_{\mathcal{A}}$) distinct indices appearing in the sampled codeword. k is the number of error-free indices we want to have in the permuted-codes sample we are constructing. I' is the set of indices left error-free by the substitution channel; we chose m so that I will be larger than I' with overwhelming probability.

We first show that $\mathcal{A}$ outputs "fail" with negligible probability. The expectation of k is $(1-\eta)n$, where $(1-\eta)$ is a constant less than δ. There is some constant δ' such that $(1-\eta) < \delta' < \delta$ and by standard binomial tail bounds, $k \leq \delta' n$ with overwhelming probability in n (and therefore in λ). Now, we need to show that I has size at least $\delta' n$. I is the set of distinct indices in $[n]$ that were included the m samples that were chosen with replacement. Since $m = \delta n$, it follows from a Chernoff bound that $|I| \geq \delta' n$ with overwhelming probability.

Provided that $\mathcal{A}$ did not output "fail," we argue that this algorithm indeed converts samples as desired. If $\mathcal{A}$ was given samples from $\mathcal{D}^{\mathsf{pp}}_{n,\mathbb{F}_q,C,T,m}$, this distribution is exactly $\mathcal{D}^{\mathsf{pc}}_{n,\mathbb{F}_q,C,\eta,T}$. That is, $\mathcal{A}$ has learned which symbols correspond to the same index i and has applied a random permutation π_i to those symbols. It has then permuted the indices under a random permutation $\pi_{\mathcal{A}}$. Similarly, if $\mathcal{A}$ was given samples from $\mathcal{D}^{\mathsf{pp}}_{n,\mathbb{F}_q,\mathbb{F}_q^n,T,m}$, its output is distributed according to $\mathcal{D}^{\mathsf{pc}}_{n,\mathbb{F}_q,\mathbb{F}_q^n,\eta,T}$. □

4.2 Statistical evidence for small alphabets

In this section we show that the permuted codes conjecture with any $\Omega(1)$ rate of random substitution errors holds for any code $C \subseteq \mathbb{F}_q^n$ with high dual distance and constant alphabet size q. This holds even without the alphabet permutations $\pi_1, \ldots, \pi_n$. Our argument is based on a decomposition of Forbes and Kelley [FK18]; we follow [HH23, Section 5.4] for our proof.

A read-once branching program (ROBP) is just a graph on vertices $V_0 \cup \cdots \cup V_n$ where each node in V_i has two edges (corresponding to 0 and 1) going to V_{i+1}. The program is evaluated on x by starting at a designated start vertex in V_0, and following the edges corresponding to the bits in x. The program outputs $f(x) = 1$ if it ends in an "accept" vertex; otherwise it outputs $f(x) = 0$.

For $v \in V_i$, we write $f_{v\to}(x)$ to denote the program evaluated on $x_{i+1}, \ldots, x_n$, starting at vertex v. We write $f_{\to v}(x)$ to denote the program evaluated on $x_1, \ldots, x_i$, where the only accept state in V_i is v.

Our main tool is the following convenient decomposition of ROBPs in the Fourier basis. It was originally stated in the work of [FK18] for the case $q = 2$, but the same statement and proof easily generalize to arbitrary q. In the following, let $\|z\|$ denote the number of non-zero coordinates in $z \in (\mathbb{Z}/q\mathbb{Z})^n$.

Lemma 4.1 ([FK18], adapted from [HH23])**.** *Let χ_z be the Fourier characters of $(\mathbb{Z}/q\mathbb{Z})^n$, defined by $\chi_z(x) = e^{2\pi i\langle x,z\rangle/q}$. Let $f : (\mathbb{Z}/q\mathbb{Z})^n \to \{0,1\}$ be a length-n width-w standard-order ROBP with layers $V_0, V_1, \ldots, V_n$. Then for any $k \in [n]$ we have*

$$f(x) = L_k(x) + H_k(x),$$

where

$$L_k(x) = \sum_{z \in (\mathbb{Z}/q\mathbb{Z})^n :\, \|z\| < k} \widehat{f}(z)\, \chi_z(x),$$

and

$$H_k(x) = \sum_{i=1}^{n} \sum_{v \in V_i} H_{k,v}(x)\, f_{v\to}(x),$$

where

$$H_{k,v}(x) = \sum_{\substack{z \in (\mathbb{Z}/q\mathbb{Z})^n, \|z\| = k \\ i \in \mathrm{supp}(z) \subseteq [i]}} \widehat{f_{\to v}}(z)\, \chi_z(x).$$

Proof. The proof is identical to that found in [HH23]. Note that $i(z)$ defined there should again be the kth-smallest index appearing in $\mathrm{supp}(z)$. □

For any $\eta \in [0,1]$, let T_η be the noise operator defined by $(T_\eta f)(x) = f(\mathrm{SC}_\eta(x))$. That is,

$$(T_\eta f)(x) = \mathop{\mathbb{E}}_{y \sim N_\eta(x)}[f(y)]$$

where the distribution of $y \sim N_\eta(x)$ is as follows: for each $i \in [n]$,

$$y_i = \begin{cases} x_i & \text{with probability } 1-\eta, \text{ and} \\ \text{uniform from } [q] & \text{with probability } \eta. \end{cases}$$

Theorem 4.2. *Suppose that $\mathcal{D}$ is a $2k$-wise uniform distribution over $(\mathbb{Z}/q\mathbb{Z})^n$. If $x \sim \mathcal{D}$ and $y \leftarrow SC_\eta(x)$, then y is $nw \cdot (1-\eta)^k$-pseudorandom to any length-n width-w ROBP f, i.e.,*

$$\left| \mathop{\mathbb{E}}_{x\sim\mathcal{D}} (T_\eta f)(x) - \mathop{\mathbb{E}}_{x\leftarrow(\mathbb{Z}/q\mathbb{Z})^n} f(x) \right| \leq nw(1-\eta)^k.$$

Proof. Applying Lemma 4.1,

$$\begin{aligned}
&\left| \mathop{\mathbb{E}}_{x\sim\mathcal{D}} [(T_\eta f)(x)] - \mathop{\mathbb{E}}_{x\leftarrow(\mathbb{Z}/q\mathbb{Z})^n} [f(x)] \right| \\
&= \left| \mathop{\mathbb{E}}_{x\sim\mathcal{D}} [(T_\eta L_k)(x) + (T_\eta H_k)(x)] - \hat{f}(\emptyset) \right| \\
&= \left| \mathop{\mathbb{E}}_{x\sim\mathcal{D}} (T_\eta H_k)(x) \right| \\
&= (1-\eta)^k \cdot \left| \mathop{\mathbb{E}}_{x\sim\mathcal{D}} \sum_{i\in[n]} \sum_{v\in V_i} (T_\eta f_{v\to})(x) \sum_{\substack{z\in(\mathbb{Z}/q\mathbb{Z})^n, \|z\|=k \\ i\in\mathrm{supp}(z)\subseteq[i]}} \hat{f}_{\to v}(z)\chi_z(x) \right| \\
&\leq (1-\eta)^k nw \cdot \max_{i,v} \mathop{\mathbb{E}}_{x\sim\mathcal{D}} \left| (T_\eta f_{v\to})(x) \sum_{\substack{z\in(\mathbb{Z}/q\mathbb{Z})^n, \|z\|=k \\ i\in\mathrm{supp}(z)\subseteq[i]}} \hat{f}_{\to v}(z)\chi_z(x) \right| \\
&\leq (1-\eta)^k nw \cdot \max_{i,v} \mathop{\mathbb{E}}_{x\sim\mathcal{D}} \left| \sum_{\substack{z\in(\mathbb{Z}/q\mathbb{Z})^n, \|z\|=k \\ i\in\mathrm{supp}(z)\subseteq[i]}} \hat{f}_{\to v}(z)\chi_z(x) \right|.
\end{aligned}$$

The second equality follows from the fact that $x \sim \mathcal{D}$ is $2k > k$-wise uniform. Now

$$\begin{aligned}
&\mathop{\mathbb{E}}_{x\sim\mathcal{D}} \left| \sum_{\substack{z\in(\mathbb{Z}/q\mathbb{Z})^n, \|z\|=k \\ i\in\mathrm{supp}(z)\subseteq[i]}} \hat{f}_{\to v}(z)\chi_z(x) \right|^2 \\
&= \mathop{\mathbb{E}}_{x\sim\mathcal{D}} \sum_{\substack{z,z'\in(\mathbb{Z}/q\mathbb{Z})^n, \|z\|=\|z'\|=k \\ i\in\mathrm{supp}(z),\mathrm{supp}(z')\subseteq[i]}} \hat{f}_{\to v}(z)\hat{f}_{\to v}(z')^*\chi_z(x)\chi_{z'}(x)^* \\
&= \sum_{\substack{z\in(\mathbb{Z}/q\mathbb{Z})^n, \|z\|=k \\ i\in\mathrm{supp}(z)\subseteq[i]}} \left| \hat{f}_{\to v}(z) \right|^2 \\
&\leq \mathop{\mathbb{E}}_{x\leftarrow\{-1,1\}^n} |f_{\to v}(x)|^2 \\
&\leq 1,
\end{aligned}$$

completing the proof. The second equality here follows from the fact that $x \sim \mathcal{D}$ is $2k$-wise uniform, so every term vanishes except the ones where $z = z'$. The first inequality follows from Parseval's identity. □

With Theorem 4.2 in hand, it is easy to see that randomly permuting the symbols of *any code with high dual distance and small alphabet* is sufficient to guarantee that a few noisy codewords are jointly uniform.

Corollary 4.3. *If $C \subseteq \mathbb{F}_q^n$ is any code with dual distance d and $\eta > 0$, then*

$$d_{\mathsf{TV}}(\mathcal{D}_{n,\mathbb{F}_q,C,SC_\eta,T}, \mathcal{U}_{\mathbb{F}_q^n,T}) \le nT \cdot \binom{n+q^T-1}{q^T-1} \cdot (1-\eta)^{d/2}$$

where $\mathcal{U}_{\mathbb{F}_q^n,T}$ is T uniform samples from $\mathbb{F}_q^n$. In particular, assuming $d = n^{\Omega(1)}$ and $\eta = \Omega(1)$:

- *if $q = O(1)$, then there exists $T = \Omega(\log d)$ such that the above is $\exp(-\Omega(d))$; and*
- *if $q^T = o(d/\log n)$, then the above is $\exp(-\Omega(d))$.*

Proof. Consider grouping together the symbols across the T samples in $\mathcal{D}_{n,\mathbb{F}_2,C,\mathrm{SC}_\eta,T}$ as symbols in $\Sigma = \mathbb{F}_q^T$. Because of the permutation, it suffices to show that the resulting string in Σ^n has the same multinomial weight distribution as a uniformly random string in Σ^n.

There are $\binom{n+q^T-1}{q^T-1}$ possibilities for what the multinomial weight (i.e., the collection of symbols and their counts) could be. Therefore we can compute it using a length-nT, width-$\binom{n+q^T-1}{q^T-1}$ ROBP. The main result then follows from applying Theorem 4.2. Finally, the special cases listed follow straightforwardly from the main result. □

Further consequence: relation to low-degree conjecture. Of independent interest, we discuss a consequence of Theorem 4.2 for the *low-degree conjecture*. For distributions P, Q on $\{0,1\}^n$, the *low-degree advantage* between them is defined, for $d \in \mathbb{N}$, as

$$\mathsf{Adv}_{\le d}(P,Q) := \max_{f:\text{deg-}k \text{ polynomial}} \frac{|\mathbb{E}_P[f] - \mathbb{E}_Q[f]|}{\sqrt{\mathbf{Var}_Q(f)}}.$$

The low-degree advantage is used to capture distinguishability by low-degree polynomials.

A special case of the *low-degree conjecture* (e.g., the $k = 1$ case of [BHJK25, Conjecture 2.1]; see also [Hop18, KWB19]) states that if a permutation-symmetric distribution is only $o(1)$ distinguishable from uniform by low-degree polynomials, then it is $o(1)$-close to uniform in statistical distance.

Formally it states the following, where we let $\mathcal{U}_n$ denote the uniform distribution on $\{0,1\}^n$:[9]

Corollary 4.4 (Special case of the low-degree conjecture)**.** *Suppose that $d_n = \omega(\log n)$ is a sequence of integers indexed by $n \in \mathbb{N}$ and that P_n is a distribution on $\{0,1\}^n$ that is invariant under any permutation. For every fixed $\eta > 0$, there is no algorithm that distinguishes between a sample from $T_\eta P_n$ and a sample from $\mathcal{U}_n$ with probability $1 - \mathsf{Adv}_{\le d_n}(P_n, \mathcal{U}_n) - o_n(1)$.*

In particular, if $\mathsf{Adv}_{\le d_n}(P_n, \mathcal{U}_n) \le o_n(1)$, then no algorithm can distinguish between the two distributions with probability $1 - o_n(1)$.

[9] Interestingly, [BHJK25], which is concurrent and independent work from the present paper, shows that the low-degree conjecture is false for $k > 1$.

Proof of Corollary 4.4. Fix $n \in \mathbb{N}$. A slight variant of Theorem 4.2 with $q = 2$, $k = d_n/2$, gives the following: for any length-n width-w ROBP f,

$$\left| \mathop{\mathbb{E}}_{x \sim P_n} (T_\eta f)(x) - \mathop{\mathbb{E}}_{x \sim \mathcal{U}_n} f(x) \right| \leq nw(1-\eta)^{d_n/2} + \mathsf{Adv}_{\leq d_n}(P_n, \mathcal{U}_n) \cdot \sqrt{\mathbf{Var}_{\mathcal{U}_n}(L_k)}. \tag{1}$$

To establish the above, we cannot directly apply Theorem 4.2, since P_n is not d_n-wise uniform. However, we instead remark that the only place where Theorem 4.2 uses that the distribution P_n is d_n-wise uniform is to establish that $\mathbb{E}_{x \sim P_n}[(T_\eta L_k)(x) - \hat{f}(0)] = 0$. Instead, we have by the fact that L_k and thus $T_\eta L_k$ is of degree at most $k < d_n$ that

$$\left| \mathop{\mathbb{E}}_{x \sim P_n} [(T_\eta L_k)(x) - \hat{f}(0)] \right| = \mathsf{Adv}_{\leq d_n}(P_n, \mathcal{U}_n) \cdot \sqrt{\mathbf{Var}_{\mathcal{U}_n}(L_k)},$$

which yields (1) by propagating the above quantity through the proof.

Next, Parseval's equality gives that $\mathbb{E}_{x \sim \mathcal{U}_n}[L(x)^2] = \sum_{z \in \{0,1\}^n} \hat{L}(z)^2 = \sum_{z \in \{0,1\}^n : \|z\| \leq k} \hat{f}(z)^2 \leq 1$. Moreover, for $d_n = \omega(\log n)$ we may apply the argument of Corollary 4.3 with $q = 2$ and $T = 1$ (corresponding to a single sample), so that the width w of the ROBP f may be taken to be $w = O(n)$, which yields

$$d_{\mathsf{TV}}(T_\eta P_n, \mathcal{U}_n) \leq O(n^2) \cdot (1-\eta)^{d_n/2} + \mathsf{Adv}_{\leq d_n}(P_n, \mathcal{U}_n) \leq o_n(1) + \mathsf{Adv}_{\leq d_n}(P_n, \mathcal{U}_n).$$

□

We remark that a variant of Corollary 4.4 which shows that no algorithm can distinguish with advantage $1 - o_n(1)$ whenever $\mathsf{Adv}_{\leq d_n}(P_n, \mathcal{U}_n) \leq O(1)$ (i.e., for any constant $O(1)$) may be obtained by using the variational characterization

$$\chi^2(P_n, \mathcal{U}_n) = \sup_{f : \{0,1\}^n \to \mathbb{R}} \frac{(\mathbb{E}_{P_n}[f] - \mathbb{E}_{\mathcal{U}_n}[f])^2}{\mathbf{Var}_{\mathcal{U}_n}(f)}$$

of the χ^2-divergence.

4.3 The alphabet permutation is necessary

Let $\mathcal{E} : \Sigma^* \to (\Sigma \cup \{\perp\})^*$ be some error channel, and $T \in \mathbb{N}$ be an integer denoting the number of samples. We define[10] $\mathcal{D}^{\mathsf{nap}}_{n,\Sigma,C,\mathcal{E},T}$ as follows:

1. Sample random permutation[11] $\sigma \leftarrow S_{[n]}$.
2. Repeat T times:
   (a) Sample $c \leftarrow C$.
   (b) Define $\hat{c}$ by $\hat{c}_i \leftarrow c_{\sigma(i)}$ for $i \in [n]$.
   (c) Output $\mathcal{E}(\hat{c})$.

[10] nap for No Alphabet Permutation.

[11] In contrast, the structured distribution considered in the permuted codes assumption (Definition 3.1) involved additionally sampling random permutations $\pi_1, \ldots, \pi_n \leftarrow S_\Sigma$. Permutation π_i was applied to the ith codeword symbol.

We demonstrate our attack on the above distribution instantiated with Reed-Solomon codes (see Definition 5.1).

Theorem 4.5. *Let C be the Reed-Solomon code $\mathsf{RS}_{\mathbb{F}_q,n,k}$ with $k \le n^{1-c}$ for some constant $c \in (0,1)$. Let $\mathcal{E} = SC_\eta$ be the substitution channel over Σ^* with some error probability $\eta \le 1-(3/2)^{-c/4}$. Then there is an efficient distinguisher achieving inverse polynomial distinguishing advantage between the distributions $\mathcal{D}_{n,\mathbb{F}_q,C,\mathcal{E},T}$ and $\mathcal{D}_{n,\mathbb{F}_q,\mathbb{F}_q^n,\mathcal{E},T}$ for $T = 4/c$.*

If $X \in \mathbb{F}^{T\times n}$, then define $X^r \in \mathbb{F}^{\binom{r+T}{r}\times n}$ to be the matrix of all coordinate-wise products of up to r rows of X, including products with repeated rows. You can think of X^r as the matrix of monomials of total degree at most r in the rows of X.

Fact 4.6. *Suppose that $p_1,\dots,p_T \in \mathbb{F}[t]$ are polynomials of degree at most k, and $t_1,\dots,t_\ell \in \mathbb{F}$. If $\bar{X}$ is the matrix where $\bar{X}_{i,j} = p_i(t_j)$ and r is any positive integer, then $\operatorname{rank}(\bar{X}^r) \le kr+1$.*

Proof. The row span of $\bar{X}^r$ consists of evaluations of polynomials of degree at most kr. But the space of all polynomials of degree at most kr has dimension $kr+1$. Therefore the row span of $\bar{X}^r$ has dimension at most $kr+1$. □

We'll apply this fact with $\bar{X}$ being the "clean" submatrix of X—that is, the submatrix of columns that don't contain any errors.

Fact 4.7. *If X is a uniformly random matrix from $\mathbb{F}^{T\times n}$ and $r^n \cdot q^{\binom{r+T}{r}-n} \ge 1$, then*

$$\mathbb{E}[\operatorname{rank}(X^r)] \ge n\cdot\left(1-\frac{\ln r}{\ln q}\right)-\frac{1}{\ln q}.$$

Proof. Let $q = |\mathbb{F}|$ and $M = \binom{r+T}{r}$. For $\alpha \in \mathbb{F}^{\binom{r+T}{r}} \setminus \{0\}$, each coordinate of αX^r is a polynomial of degree at most r in a disjoint set of coordinates of X. Therefore by Schwartz-Zippel, we have

$$\Pr_{X\leftarrow\mathbb{F}^{T\times n}}[\alpha X^r = 0] \le \left(\frac{r}{q}\right)^n.$$

Therefore the expected number of linear dependencies among rows of X^r is at most $1+r^nq^{M-n}$, so

$$\begin{aligned}
\mathbb{E}[\operatorname{rank}(X^r)] &= \mathbb{E}[M-\log_q(\#\text{ row dependencies})]\\
&\ge M-\log_q\mathbb{E}[\#\text{ row dependencies}]\\
&\ge M-\log_q\left(1+r^nq^{M-n}\right)\\
&\ge M-\log_q\left(r^nq^{M-n}\right)-\frac{1}{\ln q}\\
&= n\cdot\left(1-\frac{\ln r}{\ln q}\right)-\frac{1}{\ln q}.
\end{aligned}$$

□

We are now ready to prove the main result of this section.

Proof of Theorem 4.5. Collect our T samples into a matrix $X \in \mathbb{F}_q^{T\times n}$. Our distinguisher will simply compute $\operatorname{rank}(X^r)$ for $r = n^{c/2}$. Let us compute the expected rank in both cases.

The Reed-Solomon case. In the Reed-Solomon case where $X \leftarrow \mathcal{D}_{n,\mathbb{F}_q,C,\mathcal{E},T}$, we have $X = \bar{X}+E$ where E is all the noise from $\mathcal{E}$ and $\bar{X}$ has no noise. Fact 4.6 says that $\operatorname{rank}(\bar{X}^r) \le kr+1$. Since each column of E contains an error with probability $1-(1-\eta)^T$, the expected number of columns containing any noise is at most $n - n\cdot(1-\eta)^T$. Therefore

$$\mathbb{E}[\operatorname{rank}(X^r)] \le n - n\cdot(1-\eta)^T + kr + 1$$

in the Reed-Solomon case.

The random case. In the random case where $X \leftarrow \mathcal{D}_{n,\mathbb{F}_q,\mathbb{F}_q^n,\mathcal{E},T}$, Fact 4.7 says that

$$\mathbb{E}[\operatorname{rank}(X^r)] \ge n\cdot\left(1-\frac{\ln r}{\ln q}\right) - \frac{1}{\ln q}$$

assuming that $r^n \cdot q^{\binom{r+T}{T}-n} \ge 1$. This condition holds because $\binom{r+T}{T} \ge r^T = n^2$.

Putting it together. Since the rank is bounded by $\binom{r+T}{T} \le n^{O(1)}$, it suffices to show that these expectations differ by 1. Indeed, using the given bounds on k, η, T,

$$\begin{aligned} n - n\cdot(1-\eta)^T + kr + 1 &\le n - n\cdot((3/2)^{-c/4})^{4/c} + n^{1-c/2} + 1 \\ &= n/3 + n^{1-c/2} + 1 \end{aligned}$$

and

$$\begin{aligned} n - n\cdot\frac{\ln r}{\ln q} - \frac{1}{\ln q} &\ge n - n\cdot\frac{c}{2}\cdot\frac{\ln n}{\ln n} - \frac{1}{\ln n} \\ &\ge n/2 - 1. \end{aligned}$$

This concludes the proof. □

This attack falls into a class of *algebraic distinguishing attacks* following Arora-Ge [AG11], where one attempts to find a low-degree polynomial $P : \mathbb{F}_q^{nT} \to \mathbb{F}_q$ annihilating the given vectors. That is, given $\mathbf{v}_1, \ldots, \mathbf{v}_T$ where each $\mathbf{v}_i \in \mathbb{F}_q^n$, one attempts to find a nonzero "simple" polynomial P, such that $P(\mathbf{v}_1, \ldots, \mathbf{v}_T) = 0$. If the $\mathbf{v}_i$'s are random, such a polynomial should not exist. For certain structured distributions of interest (such as ours), P does exist, resulting in a successful distinguisher. We refer the reader to [BCH+25, Section 7] for further discussion of such attacks.

4.4 A quasipolynomial-time distinguisher for a natural class of constructions

In this section, we show that any PRC of a certain form is vulnerable to quasipolynomial-time attacks. The form is fairly general, and encompasses several prior constructions [CG24, GM24]: each PRC codeword contains a uniformly random string $x \in \{0,1\}^*$, and some predicate $f(x)$. The predicate is noise-tolerant in that if $x' \leftarrow \mathrm{SC}_\rho(x)$, $f(x') = f(x)$ with probability significantly greater than $1/2$.

We show in a formal sense that *any* such construction relies on a planted $O(\log n)$-size structure, regardless of the choice of f, or how x and f are interspersed in a longer codeword. Of course, the constructions in this paper are not of this form.

Theorem 4.8. *Consider any PRC where codewords are of length n, and of the form $\sigma(x||f(x)||y)$.*[12] *Here, $\sigma \in S_{[n]}$ is any permutation fixed across all codewords, $x \leftarrow \{0,1\}^\ell$, $f : \{0,1\}^\ell \to \{0,1\}$, and y any string in $\{0,1\}^{n-\ell-1}$ (possibly depending on x). Suppose that for a constant $\rho \in (0, 1/2)$,*

$$\Pr_{\substack{x \leftarrow \{0,1\}^\ell \\ x' \leftarrow N_\rho(x)}}[f(x') = f(x)] \geq \frac{1}{2} + \frac{1}{q}$$

for some $q = O(\text{poly}(n))$.

Then there exists an algorithm $\mathcal{A}$ running in time $n^{O(\log n)}$, distinguishing between the uniform distribution and the distribution of codewords with non-negligible advantage.

Proof. We'll argue that there is a small set S for which $\hat{f}(S) = n^{-O(\log n)}$. Estimating $\hat{f}(S)$ for all small S will yield our quasipolynomial-time distinguisher.

First, recall (see, e.g., [O'D03]) that

$$\Pr_{\substack{x \leftarrow \{-1,1\}^\ell \\ x' \leftarrow N_\rho(x)}}[f(x') = f(x)] = 1 - \mathbf{NS}_{\rho/2}(f),$$

where

$$\mathbf{NS}_{\rho/2}(f) = \frac{1}{2} - \frac{1}{2}\sum_{S \subseteq [\ell]} (1-\rho)^{|S|}\hat{f}(S)^2.$$

Therefore,

$$\frac{1}{2}\sum_{S \subseteq [\ell]} (1-\rho)^{|S|}\hat{f}(S)^2 \geq 1/q. \tag{2}$$

Now, let $t = \log_{(1-\rho)} \frac{1}{q} = O(\log n)$ and consider breaking the sum up as follows:

$$\begin{aligned}
\frac{1}{2}\sum_{S \subseteq [\ell]} (1-\rho)^{|S|}\hat{f}(S)^2 &= \frac{1}{2}\sum_{\substack{S \subseteq [\ell] \\ |S| \leq t}} (1-\rho)^{|S|}\hat{f}(S)^2 + \frac{1}{2}\sum_{\substack{S \subseteq [\ell] \\ |S| > t}} (1-\rho)^{|S|}\hat{f}(S)^2 \\
&\leq \frac{1}{2}\binom{\ell}{\leq t}\max_{\substack{S \subseteq [\ell] \\ |S| \leq t}} \hat{f}(S)^2 + \frac{1}{2}(1-\rho)^{|S|} \text{ (Parseval's identity)} \\
&\leq \frac{1}{2}(\ell^t + 1)\max_{\substack{S \subseteq [\ell] \\ |S| \leq t}} \hat{f}(S)^2 + \frac{1}{2q}.
\end{aligned}$$

In order for this quantity to be at least $1/q$, we must have

$$\max_{\substack{S \subseteq [\ell] \\ |S| \leq t}} \hat{f}(S)^2 \geq \frac{1}{(\ell^t + 1) \cdot q}, \text{ where } \ell \leq n \text{ and } t = O(\log n).$$

Recall our task of distinguishing between strings of the form $\pi(x, f(x), y)$ and $r \leftarrow \{0,1\}^n$. In the latter case, for all $S \subseteq [n]$ and all $i \notin S$, $\mathbb{E}_{r \leftarrow \{-1,1\}^n}[\chi_S(r) \cdot r_i] = 0$ (mapping r to $\{-1,+1\}^n$). In the former case, we just showed that there is a set S of size $O(\log n)$, and an index i, such that this expectation is $1/n^{O(\log n)}$. Therefore, there is a distinguisher running in time $n^{O(\log n)}$ that estimates this expectation for all sets S of size at most t, and all i. □

[12] Abusing notation a bit, we mean that the ith bit of the codeword is the $\sigma(i)^{\text{th}}$ bit of $(x||f(x)||y)$.

5 Edit-robust pseudorandom codes

In this section, we present our main technical consequence of the permuted codes assumption, namely edit-robust PRCs over a binary alphabet with strong adaptive robustness (and subexponential security, assuming such for the permuted codes assumption). In Section 5.1 we present some preliminaries from coding theory, and in Section 5.2 we do the same for edit distance. In Sections 5.3 and 5.4 we present our main PRC construction and its analysis, and in Section 5.5 we show how to derive analogous guarantees for watermarking from our PRC.

5.1 Preliminaries on folded Reed-Solomon codes

Consider a field $\mathbb{F}_q$, and integers s, n with $n \leq q-1$ such that n is divisible by s. Let γ be a generator of $\mathbb{F}_q^\star$, and $k \in \mathbb{N}$.

Definition 5.1 (Reed-Solomon code)**.** The *Reed-Solomon code* $\mathsf{RS}_{\mathbb{F}_q,n,k}$ is a linear code of dimension $k+1$ and block length n over $\mathbb{F}_q$. It consists of the vectors of evaluations of polynomials of degree at most k with coefficients in $\mathbb{F}_q$ on a set of distinct elements $x_1, \ldots, x_n \in \mathbb{F}_q$:

$$\mathsf{RS}_{\mathbb{F}_q,n,k} = \{(p(x_1), \ldots, p(x_n)) \mid p \in \mathbb{F}_q[x], \deg(p) \leq k\}.$$

Definition 5.2 (Folded Reed-Solomon code)**.** The *folded Reed-Solomon code* $\mathsf{FRS}_{\mathbb{F}_q,\gamma,n,s,k}$ is a code of block length $N := n/s$ over $\mathbb{F}_q^s$. For a message $p(X)$, a polynomial over $\mathbb{F}_q$ of degree at most k, for $0 \leq j < N$, the jth symbol of the encoding is $(p(\gamma^{js}), p(\gamma^{js+1}), \ldots, p(\gamma^{js+s-1}))$.

We will need to use the fact that (folded) Reed-Solomon codes have very good *list recovery* algorithms, defined below.

Definition 5.3 (List recovery)**.** Fix $\zeta \in [0,1]$ and $\ell, L \in \mathbb{N}$. A code $C \subset \Sigma^n$ is (ζ, ℓ, L)*-list recoverable* if for every sequence of sets $S_1, \ldots, S_n \subset \Sigma$ satisfying $|S_i| \leq \ell$ for all $i \in [n]$, there are at most L codewords $c \in C$ for which $c_i \in S_i$ for at least ζn values of $i \in [n]$.

For some $T = T(n)$, we say that C is (ζ, ℓ)*-list recoverable in time* T if there is some $L = L(n) \leq O(T(n))$ and a $O(T(n))$-time algorithm which, given $c \in \Sigma^n$, finds the at most L codewords c' for which $c'_i \in S_i$ for at least ζn values of $i \in [n]$. The *agreement threshold* is defined as $t_{\mathsf{rec}} = \zeta n$.

A classical fact in coding theory, due to [GS98], is that Reed-Solomon codes have good list recovery algorithms:

Theorem 5.1 ([GRS25], Theorem 12.3.4)**.** *Consider the Reed-Solomon code* $\mathsf{RS}_{\mathbb{F}_q,n,k}$ *as above. Then for any* $\zeta \in (0,1), \ell \in \mathbb{N}$ *satisfying*

$$\zeta n \geq \sqrt{(k-1)\ell n},$$

we can (ζ, ℓ)*-list recover the code* $\mathsf{RS}_{\mathbb{F}_q,n,k}$ *in time* $\mathrm{poly}(n)$*; that is, there exists an efficient list recovery algorithm with* $t_{\mathsf{rec}} = \zeta n$*.*

It is possible to obtained improved list recovery algorithms by considering folded Reed-Solomon codes (which come with the caveat that their alphabet is increased to size q^s). In the below theorem, we fix a field $\mathbb{F}_q$ and integers n, s, k, γ parametrizing a folded Reed-Solomon code $\mathsf{FRS}_{\mathbb{F}_q,\gamma,n,s,k}$ as in Definition 5.2; thus its block length is $N = n/s$.

Theorem 5.2 ([GR07], Theorem 4.4 & Eq. (10))**.** *Consider the folded Reed-Solomon code* $\mathsf{FRS}_{\mathbb{F}_q,\gamma,n,s,k}$ *as above. Then for any* $\zeta \in (0,1), \ell \in \mathbb{N}$ *satisfying*

$$\zeta N \geq k \cdot (N\ell)^{1/(s+1)} + 2,$$

we can (ζ,ℓ)*-list recover the code* $\mathsf{FRS}_{\mathbb{F}_q,\gamma,n,s,k}$ *in time* $N^{O(s)}$*.*

Pseudorandomness of permuted FRS codes. In order to obtain PRCs with a binary alphabet robust to some constant fraction of edits, it will suffice for us to use the permuted codes conjecture for Reed-Solomon codes (Conjecture 3.2). However, in order to obtain an analogous *watermarking scheme* that works for language models whose per-token entropy is an arbitrary constant, we shall need to slightly modify our PRC construction, to use *folded* Reed-Solomon codes. As such codes are not linear over their alphabet, we cannot directly apply the permuted codes assumption of Definition 3.1 to argue that randomly permuted codewords are pseudorandom. Nevertheless, such codes are linear over $\mathbb{F}_q$, and they have high dual distance with respect to the linear structure over $\mathbb{F}_q$, in the sense that for a uniformly random codeword, its marginal distribution on any k/s symbols is uniform. Therefore, we believe the following generalization of Conjecture 3.2 is plausible for such codes:

Conjecture 5.3 (Permuted FRS conjecture)**.** *Fix any constant* $\eta > 0$*. Let* λ *be a security parameter and* C *be the folded Reed-Solomon code* $\mathsf{FRS}_{\mathbb{F}_q,\gamma,n,s,k}$ *with*

- γ *any generator of* $\mathbb{F}_q$*,*
- $n = \lambda$*,*
- $q = \lambda + 1$*,*
- $s = O(1)$ *any constant, and*
- $k = \lambda^{1/(s+1)}$*.*

Writing $r = q^s$*, we have that the distributions* $\mathcal{D}_{n,[r],C,\eta,T}$ *and* $\mathrm{Unif}(([r]^{n/s})^T)$ *are computationally indistinguishable for any* $T = \mathrm{poly}(\lambda)$*.*

5.2 Preliminaries on edit distance

Below we present some basic preliminaries regarding edit-distance bounded channels.

Definition 5.4. Given strings $z, z' \in \Sigma^n$, the *edit distance* between z and z', denoted $D_{\mathsf{E}}(z,z')$, is defined as the minimum number of insertions and deletions needed to transform z into z'.

The Hamming distance is denoted $D_{\mathsf{H}}(z,z')$.

Definition 5.5. Given $z \in \Sigma^n$ and a real number $p \in [0,1]$, we let

$$\mathcal{B}_{\mathsf{E}}(z,p) := \{z' \in \Sigma^\star \ : \ D_{\mathsf{E}}(z,z') \leq p \cdot |z|\}$$

denote the edit distance ball of radius $p \cdot |z|$ centered at z. Given real numbers $p_{\mathsf{H}}, p_{\mathsf{E}} \in [0,1]$, we let

$$\mathcal{B}_{\mathsf{H},\mathsf{E}}(z,p_{\mathsf{H}},p_{\mathsf{E}}) := \{z' \in \Sigma^\star \ : \ \exists z'' \in \Sigma^\star, \ D_{\mathsf{H}}(z,z'') \leq p_{\mathsf{H}} \cdot |z|, D_{\mathsf{E}}(z'',z') \leq p_{\mathsf{E}} \cdot |z|\}$$

denote the set of strings into which z can be transformed by first making $p_{\mathsf{H}} \cdot |z|$ substitutions and then making $p_{\mathsf{E}} \cdot |z|$ edits.

Definition 5.6 (Edit-bounded channel)**.** Fix real numbers $\varepsilon_{\mathsf{H}}, \varepsilon_{\mathsf{E}} \in [0,1]$. A mapping (i.e., "channel") $\mathcal{E} : \Sigma^\star \to \Sigma^\star$ is defined to be ε_{E}-*edit bounded* if for all $z \in \Sigma^\star$, $\mathcal{E}(z) \in \mathcal{B}_{\mathsf{E}}(z, \varepsilon_{\mathsf{E}})$. We denote the set of ε_{E}-edit bounded channels by $\mathscr{E}^{\mathsf{E}}_{\varepsilon_{\mathsf{E}}}$.

A channel $\mathcal{E} : \Sigma^\star \to \Sigma^\star$ is defined to be $(\varepsilon_{\mathsf{H}}, \varepsilon_{\mathsf{E}})$-*substitution-edit bounded* if for all $z \in \Sigma^\star$, $\mathcal{E}(z) \in \mathcal{B}_{\mathsf{H},\mathsf{E}}(z, \varepsilon_{\mathsf{H}}, \varepsilon_{\mathsf{E}})$. We denote the set of $(\varepsilon_{\mathsf{H}}, \varepsilon_{\mathsf{E}})$-substitution-edit bounded channels by $\mathscr{E}^{\mathsf{H},\mathsf{E}}_{\varepsilon_{\mathsf{H}},\varepsilon_{\mathsf{E}}}$.

Lemma 5.4. *For any $z \in \Sigma^n$ and $d \in \mathbb{N}$, $|\mathcal{B}_{\mathsf{E}}(z, d/n)| \le \left(\frac{e(n+d)\cdot(|\Sigma|+1)}{d}\right)^d$. In particular, if $d = \epsilon n$ for some $\epsilon \in (0,1)$, then we have $|\mathcal{B}_{\mathsf{E}}(z,\epsilon)| \le (2e(|\Sigma|+1))^{n\epsilon\log(1/\epsilon)}$.*

Proof. There are at most $\binom{n+d}{d}$ ways to choose the positions at which to make insertions and deletions; for each one, one can either make a deletion or an insertion of any of $|\Sigma|$ characters. Overall, the number of ways to make a sequence of $n+d$ edits is thus bounded above by

$$\binom{n+d}{d} \cdot (|\Sigma|+1)^d \le \left(\frac{e(n+d)\cdot(|\Sigma|+1)}{d}\right)^d,$$

which, in the case that $d = \epsilon n$ for some $\epsilon \in (0,1)$, may be bounded above by $(2e(|\Sigma|+1))^{n\epsilon\log(1/\epsilon)}$. □

Lemma 5.5. *For any $z \in \{0,1\}^n$ and $p, \epsilon \in (0,1)$, we have $|\mathcal{B}_{\mathsf{H},\mathsf{E}}(z, 1/2-p, \epsilon)| \le 2^{n\cdot(1-p^2+\log(6e)\epsilon\log(1/\epsilon))}$.*

Proof. The number of z'' for which $D_{\mathsf{H}}(z,z'') \le (1/2-p)\cdot n$ may be bounded above by $2^{n\cdot h_2(1/2-p)} \le 2^{n\cdot(1-p^2)}$, where $h_2(\cdot)$ denotes the binary entropy. Using Lemma 5.4 with $|\Sigma| = 2$ we see that

$$|\mathcal{B}_{\mathsf{H},\mathsf{E}}(z,(1/2-p),\epsilon)| \le 2^{n\cdot(1-p^2)} \cdot (6e)^{n\epsilon\log(1/\epsilon)} = 2^{n\cdot(1-p^2+\log(6e)\epsilon\log(1/\epsilon))}.$$

□

5.3 Our PRC construction: Algorithm 1

Algorithm 1 shows our main PRC construction. It takes as input a code $C(\lambda) \subset [q(\lambda)]^{n(\lambda)}$ (for a security parameter λ), as well as additional parameters $p_{\mathsf{Dec}}, \varepsilon_{\mathsf{Dec}}, L_{\mathsf{max}}, \eta, t_{\mathsf{rec}}$, as discussed further below. The key generation algorithm $\mathsf{KeyGen}(1^\lambda)$ takes as input the security parameter λ and returns a key $\mathsf{sk} = (\sigma, \pi_1, \ldots, \pi_n, o)$ consisting of permutations $\sigma, \pi_1, \ldots, \pi_n$ used to permute codewords, as well as a one-time pad o (which is used to ensure soundness of the PRC).

The encoding algorithm $\mathsf{Encode}(1^\lambda, \mathsf{sk})$ draws a uniformly random codeword from C, and then permutes it according to sk and adds noise η in a way consistent with the permuted codes assumption (Conjecture 3.1). Finally, it outputs a subset of the symbols in the resulting (permuted and noisy) codeword, where each symbol is accompanied by its index, and both are written in binary.

Finally, the decoding algorithm $\mathsf{Decode}(1^\lambda, \mathsf{sk}, y)$ takes as input the key sk and a string y and applies a list recovery algorithm (using the parameters $L_{\mathsf{max}}, t_{\mathsf{rec}}, p_{\mathsf{Dec}}, \varepsilon_{\mathsf{Dec}}$) to determine if y is close to some codeword in $C(\lambda)$. We refer the reader to Section 3.2 for further intuitive explanations regarding the design of the encoding and decoding algorithms.

Algorithm 1 Edit-robust PRC with binary alphabet

Require: Functions $q = q(\lambda), n = n(\lambda), k = k(\lambda), C = C(\lambda) \subset [q]^n$ denoting alphabet size, block length, code dimension, and the code, respectively (all a function of the security parameter λ). We assume $n(\lambda), q(\lambda)$ are powers of 2 for all λ. *Additional parameters:* $m, p_{\mathsf{Dec}}, \varepsilon_{\mathsf{Dec}}, L_{\mathsf{max}}, \eta$, and a list recovery algorithm $\mathsf{ListRecovery}$ for C with agreement threshold t_{rec}.

function $\mathsf{KeyGen}(1^\lambda)$
  Write $n = n(\lambda)$.
  Let $\sigma \leftarrow \mathrm{Unif}(S_{[n]}), \pi_1, \ldots, \pi_n \leftarrow \mathrm{Unif}(S_{[q]})$ be chosen uniformly at random.
  Choose $o \sim \mathrm{Unif}([q]^n)$.
  return $(\sigma, \pi_1, \ldots, \pi_n, o)$.
function $\mathsf{Encode}(1^\lambda, \mathsf{sk})$
  Write $\mathsf{sk} = (\sigma, \pi_1, \ldots, \pi_n, o)$.
  Draw $c \leftarrow \mathrm{Unif}(C)$ and write $c' \leftarrow \mathrm{SC}_\eta(c)$. *($SC_\eta$ is substitution channel w/ noise rate η.)*
  Set $z \leftarrow c' + o \pmod q \in [q]^n$.
  Sample i.i.d. indices $i_1, \ldots, i_m \leftarrow \mathrm{Unif}([n])$.
  for $j = 1, 2, \ldots, m$ **do**
    if $i_j \neq i_{j'} \forall j' < j$ **then**
      Define $z'_j := \pi_{i_j}(z_{\sigma(i_j)})$.
    else
      Sample $z'_j \leftarrow \mathrm{Unif}([q])$.
  Let y denote the concatenation: $\mathsf{bin}(i_1) \circ \mathsf{bin}(z'_1) \circ \mathsf{bin}(i_2) \circ \mathsf{bin}(z'_2) \circ \ldots \circ \mathsf{bin}(i_m) \circ \mathsf{bin}(z'_m)$.
  return $y \in \{0,1\}^{m \cdot (\log q + \log n)}$.
function $\mathsf{Decode}(1^\lambda, \mathsf{sk}, y)$
  Write $\mathsf{sk} = (\sigma, \pi_1, \ldots, \pi_n, o)$, $\ell := \log q + \log n$.
  For each $j \in [n]$, initialize $\mathcal{L}_j \leftarrow \emptyset$.
  for $1 \leq j \leq |y| - \ell$ **do**
    For each pair $(i, z) \in [n] \times [q]$ for which the below holds, add $\pi_i^{-1}(z)$ to $\mathcal{L}_{\sigma(i)}$:

$$y_{j:j+\ell-1} \in \mathcal{B}_{\mathsf{H},\mathsf{E}}(\mathsf{bin}(i) \circ \mathsf{bin}(z), 1/2 - p_{\mathsf{Dec}}, \varepsilon_{\mathsf{Dec}}).$$

  For each $i \in [n]$ with $|\mathcal{L}_i| \geq L_{\mathsf{max}}$, remove $|\mathcal{L}_i| - L_{\mathsf{max}}$ arbitrary elements of $\mathcal{L}_i$.
  if $\mathsf{ListRecovery}(t_{\mathsf{rec}}, (\mathcal{L}_i - o_i)_{i \in [n]}) \neq \emptyset$ **then** *($\mathcal{L}_i - o_i$ indicates shifting $\mathcal{L}_i$ by $-o_i$.)*
    return True
  else
    return False

Parameter settings. We consider two distinct settings of parameters for our edit-robust pseudorandom codes. The first setting of parameters is used to obtain the guarantee of strong adaptive robustness with respect to any ε_{E}-edit bounded channel:

Definition 5.7 (Edit parameters)**.** Given $\varepsilon_{\mathsf{E}} > 0$, we define the following parameters of Algorithm 1, so as to obtain robustness to ε_{E}-edit distance bounded channels:

- For a security parameter λ, we choose the parameters of the code as follows:

$$n(\lambda) = \lambda, \quad q(\lambda) = \lambda - 1, \quad k(\lambda) = n^{1/5}.$$

  To simplify notation, we write $q = q(\lambda), n = n(\lambda), k = k(\lambda)$, and we let $\gamma = \gamma(\lambda)$ denote a generator of $\mathbb{F}_q$. We next define the code $C = C(\lambda)$ by $C := \mathsf{RS}_{\mathbb{F}_q,n,k}$.

- $m = 4n^{4/5}$.
- $C_{\mathsf{Dec}} = 16$.
- $\eta = 1/32$.
- $L_{\mathsf{max}} = n^{2/5}$.
- $\varepsilon_{\mathsf{Dec}} = 2\varepsilon_{\mathsf{E}} C_{\mathsf{Dec}}$, $p_{\mathsf{Dec}} = 1/2$.
- $t_{\mathsf{rec}} = \sqrt{kL_{\mathsf{max}}n}$.
- $\mathsf{ListRecovery}$ is the algorithm described in Theorem 5.1.

The second setting of parameters is used to obtain the guarantee of strong robustness with respect to any $(1/2 - p_{\mathsf{Sub}}, \varepsilon_{\mathsf{E}})$-substitution-edit bounded channel, which will in turn be needed for our watermarking application in the Section 5.5.

Definition 5.8 (substitution-edit parameters)**.** Given $p_{\mathsf{Sub}} \in (0, 1/2)$, we define the following parameters for Algorithm 1, so as to obtain robustness to $(1/2 - p_{\mathsf{Sub}}, \varepsilon_{\mathsf{E}})$-substitution-edit distance bounded channels, for an appropriate choice of ε_{E} (specified below):

- $s = 8/p_{\mathsf{Sub}}^2$ (representing the folding parameter for the FRS code).
- For a security parameter λ, we choose the parameters of the code as follows:

$$\bar{n}(\lambda) = \lambda, \quad \bar{q}(\lambda) = \lambda - 1, \quad k(\lambda) = n(\lambda)^{1/(s+1)}, \quad n(\lambda) = \bar{n}(\lambda)/s, \quad q(\lambda) = \bar{q}(\lambda)^s.$$

  To simplify notation, we write $q = q(\lambda)n = n(\lambda), k = k(\lambda)$, and we let $\gamma = \gamma(\lambda)$ denote a generator of $\mathbb{F}_q$. We next define the code $C = C(\lambda)$ by $C := \mathsf{FRS}_{\mathbb{F}_q,\gamma,n,s,k}$. Above $\bar{n}(\lambda), \bar{q}(\lambda)$ should be interpreted as the block length and alphabet size of the underlying Reed-Solomon code.

- $m = n^{s/(s+1)}$.
- $C_{\mathsf{Dec}} = 16/p_{\mathsf{Sub}}$.
- $\eta = p_{\mathsf{Sub}}/32$.

- $L_{\mathsf{max}} = n^{s-3}$.
- $\varepsilon_{\mathsf{E}} = c_0 \cdot p_{\mathsf{Sub}}^3 \log(1/p_{\mathsf{Sub}})$, where c_0 is a universal constant chosen sufficiently small so that $\varepsilon_{\mathsf{E}} \log(1/\varepsilon_{\mathsf{E}}) \log(6e) \leq p_{\mathsf{Sub}}^3/32$.
- $p_{\mathsf{Dec}} = p_{\mathsf{Sub}}/2$, $\varepsilon_{\mathsf{Dec}} = 2\varepsilon_{\mathsf{E}} C_{\mathsf{Dec}}$. By our choice of ε_{E} and c_0 above, it follows that $\varepsilon_{\mathsf{Dec}} \log(6e) \log(1/\varepsilon_{\mathsf{Dec}}) \leq p_{\mathsf{Sub}}^2/2$.
- $t_{\mathsf{rec}} = k \cdot (nL_{\mathsf{max}})^{1/(s+1)} + 2$.

5.4 Analysis of Algorithm 1

In this section, we will establish the following theorem which shows that the PRC of Algorithm 1 obtains all of our desired properties:

Theorem 5.6 (Main edit-robust PRC result)**.** *Fix any $\varepsilon_{\mathsf{E}} \leq 1/400$. Then the PRC construction of Algorithm 1 with the parameter settings of Definition 5.7 is undetectable, sound, and has strong adaptive robustness to any ε_{E}-edit bounded channel, where undetectability relies on Conjecture 3.2. Moreover, the constituent algorithms $\mathsf{KeyGen}, \mathsf{Encode}, \mathsf{Decode}$ all run in $\mathrm{poly}(\lambda)$ time, where λ denotes the security parameter.*

Proof. Strong adaptive robustness to ε_{E}-edit bounded channels follows from Lemma 5.11. Soundness follows from Lemma 5.12. Undetectability, under Conjecture 3.2, follows from Lemma 5.13. □

Additionally, to obtain a watermarking scheme in Section 5.5, we will need the following analogous theorem which applies to any substitution-edit bounded channel with appropriate parameters:

Theorem 5.7. *Fix any $p_{\mathsf{Sub}} \in (0, 1/2)$. Then the PRC construction of Algorithm 1 with the parameter settings of Definition 5.8 is undetectable, sound, and has strong robustness to any $(1/2 - p_{\mathsf{Sub}}, \varepsilon_{\mathsf{E}})$-substitution-edit bounded channel, for $\varepsilon_{\mathsf{E}} = \tilde{\Theta}(p_{\mathsf{Sub}}^3)$, where undetectability relies on Conjecture 5.3. Moreover, the constituent algorithms $\mathsf{KeyGen}, \mathsf{Encode}$ run in $\mathrm{poly}(\lambda)$ time,[13] and Decode runs $\lambda^{O(1/p_{\mathsf{Sub}}^2)}$ time, where λ denotes the security parameter.*

Proof. Strong adaptive robustness to ε_{E}-edit bounded channels follows from Lemma 5.10. Soundness follows from Lemma 5.12. Undetectability, under Conjecture 3.2, follows from Lemma 5.13. The bound on the running time follows immediately from Theorem 5.2. □

We remark that one downside of Theorem 5.7 is that the PRC's decoding algorithm runs in time that grows exponentially in $1/p_{\mathsf{Sub}}^2$; this is due to our use of folded Reed-Solomon codes, which require time $n^{O(s)}$ to list recover (where $s = O(1/p_{\mathsf{Sub}}^2)$ is the folding parameter). It is an interesting question to obtain an improved construction which does not suffer this decay.

We proceed with the proofs of the individual lemmas used to establish Theorems 5.6 and 5.7. Our first lemma gives an upper bound on the size of the lists $\mathcal{L}_j$ constructed in the decoding algorithm Decode.

Lemma 5.8. *The lists $\mathcal{L}_j$ at termination of the for loop on Line 21 of $\mathsf{Decode}(1^\lambda, \mathsf{sk}, y)$ satisfy the following:*

$$\sum_{j=1}^{n} |\mathcal{L}_j| \leq \begin{cases} |y| \cdot (nq)^{1 - p_{\mathsf{Dec}}^2 + \log(6e)\varepsilon_{\mathsf{Dec}} \log(1/\varepsilon_{\mathsf{Dec}})} & : p_{\mathsf{Dec}} \in [0, 1/2) \\ |y| \cdot (nq)^{\varepsilon_{\mathsf{Dec}} \log(6e) \log(1/\varepsilon_{\mathsf{Dec}})} & : p_{\mathsf{Dec}} = 1/2. \end{cases}$$

[13] In particular, this $\mathrm{poly}(\lambda)$ does not depend exponentially on $1/p_{\mathsf{Sub}}^2$.

Proof. Let the input to Decode be denoted y. For each $j \in [|y| - \ell]$, the number of tuples (w, z) satisfying the conditions on Line 22 may be bounded above using Lemma 5.5 (for any $p \in [0, 1/2]$) or, in the case that $p = 1/2$, by the tighter bound in Lemma 5.4 (where the block length n in each of these lemmas is taken to be $\ell = \log(nq)$). This yields the bound claimed in the lemma statement. □

Next, we show that for any string y which is output by Encode, if it is passed through an edit-bounded channel (or more generally, an edit-substitution bounded channel), then many of the lists $\mathcal{L}_i$ constructed in Decode will still contain correct codeword symbols corresponding to the codeword generated by Encode.

Lemma 5.9. *Consider any $i_1, \ldots, i_m \in [n], z'_1, \ldots, z'_m \in [q]$, and let $y \in \{0,1\}^{m \cdot \ell}$ denote the concatenation $\mathsf{bin}(i_1) \circ \mathsf{bin}(z'_1) \circ \ldots \circ \mathsf{bin}(i_m) \circ \mathsf{bin}(z'_m)$. Fix any $y' \in \mathcal{B}_{\mathsf{H},\mathsf{E}}(y, 1/2 - p_{\mathsf{Sub}}, \varepsilon_{\mathsf{E}})$, and any value of sk. For $j \in [m]$, letting $\mathcal{L}_{i_j}$ denote the value of the list in $\mathsf{Decode}(1^\lambda, \mathsf{sk}, y')$ at the conclusion of the function, then the number of values of $j \in [m]$ for which $z'_j \in \mathcal{L}_{i_j}$ is at least*

$$m \cdot \left(\frac{p_{\mathsf{Sub}}}{2} - \frac{1}{C_{\mathsf{Dec}}} - \frac{2\ell \cdot (nq)^{1 - p_{\mathsf{Sub}}^2 + \varepsilon_{\mathsf{Dec}} \log(6e) \log(1/\varepsilon_{\mathsf{Dec}})}}{L_{\mathsf{max}}} \right) \quad \text{if} \quad p_{\mathsf{Sub}} < 1/2$$

$$m \cdot \left(1 - \frac{1}{C_{\mathsf{Dec}}} - \frac{2\ell \cdot (nq)^{\varepsilon_{\mathsf{Dec}} \log(6e) \log(1/\varepsilon_{\mathsf{Dec}})}}{L_{\mathsf{max}}} \right) \quad \text{if} \quad p_{\mathsf{Sub}} = 1/2.$$

Proof. Fix $y'' \in \{0,1\}^\star$ so that $D_{\mathsf{H}}(y, y'') \le (1/2 - p_{\mathsf{Dec}}) \cdot m\ell$ and $D_{\mathsf{E}}(y', y'') \le \varepsilon_{\mathsf{Dec}} \cdot m\ell$. For each $j \in [m]$, we define the following quantities:

- Let $f_j \ge 0$ denote the number of substitutions made at positions $(j-1)\,\ell + 1, \ldots, (j-1)\,\ell + \ell$ when transforming y to y''.
- Let $e_j \ge 0$ denote the number of edits (i.e., insertions and deletions) made at positions $(j-1)\,\ell + 1, \ldots, (j-1)\,\ell + \ell$ when transforming y'' to y' (through an optimal sequence of edits).

Using the bounds on $D_{\mathsf{H}}(y, y'')$ and $D_{\mathsf{E}}(y'', y')$, we have

$$f_1 + \cdots + f_m \le (1/2 - p_{\mathsf{Sub}}) \cdot m\ell, \qquad e_1 + \cdots + e_m \le \varepsilon_{\mathsf{E}} \cdot m\ell.$$

Then the following statements are immediate:

- For at least $m \cdot (1 - 1/C_{\mathsf{Dec}})$ values of $j \in [m]$, we have that $e_j \le \varepsilon_{\mathsf{E}} \cdot \ell C_{\mathsf{Dec}}$.
- For at least $m \cdot p_{\mathsf{Sub}}/2$ values of $j \in [m]$, we have $f_j \le (1/2 - p_{\mathsf{Sub}}/2) \cdot \ell$. If in fact $p_{\mathsf{Sub}} = 1/2$, then $f_j = 0$ for all $j \in [m]$.

Let $\mathcal{J} \subset [m]$ be the set of $j \in [m]$ for which $e_j \le \varepsilon_{\mathsf{E}} \cdot \ell C_{\mathsf{Dec}}$ and $f_j \le (1/2 - p_{\mathsf{Sub}}/2) \cdot \ell$. Then $|\mathcal{J}| \ge m \cdot (p_{\mathsf{Sub}}/2 - 1/C_{\mathsf{Dec}})$ (and $|\mathcal{J}| \ge m \cdot (1 - 1/C_{\mathsf{Dec}})$ in the case that $p_{\mathsf{Sub}} = 1/2$). For each $j \in \mathcal{J}$, there is some $j' \in |y'|$ so that

$$y'_{j:j+\ell} \in \mathcal{B}_{\mathsf{H},\mathsf{E}}(\mathsf{bin}(i_j) \circ \mathsf{bin}(z'_j), 1/2 - p_{\mathsf{Sub}}/2, 2\varepsilon_{\mathsf{E}} C_{\mathsf{Dec}}).$$

Note that the edit distance term above is $2\varepsilon_{\mathsf{E}} C_{\mathsf{Dec}}$ since an additional $e_j \leq \varepsilon_{\mathsf{E}} \cdot \ell C_{\mathsf{Dec}}$ deletions may be needed following the edits to transform y'' to y' (as in the above expression we are considering the fixed-length quantity $y'_{j:j+\ell}$).

Since $\varepsilon_{\mathsf{Dec}} \geq 2\varepsilon_{\mathsf{E}} C_{\mathsf{Dec}}$ and $p_{\mathsf{Dec}} \leq p_{\mathsf{Sub}}/2$ (or else $p_{\mathsf{Dec}} = p_{\mathsf{Sub}} = 1/2$), it follows that, for $j \in \mathcal{J}$, in Line 22 of Decode, z'_j is added to $\mathcal{L}_{i_j}$.

Let us write

$$\hat{L} := \begin{cases} |y| \cdot (nq)^{1-p_{\mathsf{Dec}}^2 + \log(6e)\varepsilon_{\mathsf{Dec}} \log(1/\varepsilon_{\mathsf{Dec}})} & : p_{\mathsf{Dec}} < 1/2 \\ |y| \cdot (nq)^{\varepsilon_{\mathsf{Dec}} \log(6e) \log(1/\varepsilon_{\mathsf{Dec}})} & : p_{\mathsf{Dec}} = 1/2. \end{cases}$$

By Lemma 5.8 and the fact that $|y'| \leq (1+\varepsilon_{\mathsf{E}}) \cdot m\ell \leq 2m\ell$, the lists $\mathcal{L}_i$ at the termination of the for loop on Line 21 of Decode satisfy $\sum_{i=1}^{n} |\mathcal{L}_i| \leq 2m\ell \cdot \hat{L}$. Thus, for $\eta := \frac{2m\ell \cdot \hat{L}}{nL_{\mathsf{max}}}$, the number of lists $\mathcal{L}_w$ for which some item is removed in Line 23 of Decode may be bounded above by $\eta \cdot n$. Overall, it follows that the number of values of $j \in [m]$ for which $z'_j \in \mathcal{L}_{i_j}$ at termination of Decode is at least the quantity claimed in the lemma statement. □

Using Lemma 5.9, we next establish the strong adaptive robustness of our PRC, for the parameter settings of Definition 5.8 (in Lemma 5.10) and Definition 5.7 (in Lemma 5.11).

Lemma 5.10. *Given the parameters of Definition 5.8, the PRC of Algorithm 1 is strongly robust to $(1/2 - p_{\mathsf{Sub}}, \varepsilon_{\mathsf{E}})$-substitution-edit distance bounded channels.*

Proof. We will show that for any value of sk,

$$\Pr_{y \leftarrow \mathsf{Encode}(1^\lambda, \mathsf{sk})} \left(\forall \mathcal{E} \in \mathscr{E}^{\mathsf{H},\mathsf{E}}_{1/2-p_{\mathsf{H}},\varepsilon_{\mathsf{E}}},\ \mathsf{Decode}(1^\lambda, \mathsf{sk}, \mathcal{E}(y)) = \mathsf{True} \right) \geq 1 - \exp\left(-n^{\Omega(1)}\right). \tag{3}$$

Fix any value for sk. Let $\mathcal{E}_{\mathsf{good}}$ be the event that the following conditions hold:

- Amongst the indices $i_1, \ldots, i_m$ generated on Line 10 of Encode, there are at least $m - 2m^2/n$ distinct elements.
- The substitution channel SC_η on Line 8 of Encode corrupts at most $2\eta m$ of the m symbols of c given by $c_{\sigma(i_1)}, \ldots, c_{\sigma(i_m)}$.

The probability that each i_j is distinct from $i_1, \ldots, i_{j-1}$ is at least $1 - (j-1)/n \geq 1 - m/n$, and thus Azuma's inequality ensures that the probability of the first item above is at least $1 - \exp\left(-\Omega(m^3/n^2)\right) \geq 1 - \exp\left(-n^{\Omega(1)}\right)$, since $m \geq n^{3/4}$. Moreover, by a Chernoff bound, the probability that the second item above fails is at most $\exp(-\Omega(\eta m)) \leq \exp\left(-n^{\Omega(1)}\right)$, since $\eta = \Omega(1)$. Overall, it follows that $\Pr(\mathcal{E}_{\mathsf{good}}) \geq 1 - \exp\left(-n^{\Omega(1)}\right)$. Let $\mathcal{I} := \{i_1, \ldots, i_m\} \subset [n]$, so that $|\mathcal{I}| \geq m - 2m^2/n$ whenever the event $\mathcal{E}_{\mathsf{good}}$ occurs.

The output of $\mathsf{Encode}(1^\lambda, \mathsf{sk})$, which we denote by $y \in \{0,1\}^{m\ell}$, is the concatenation of $\mathsf{bin}(i_1), \mathsf{bin}(z'_1), \ldots$ $\mathsf{bin}(i_m), \mathsf{bin}(z'_m)$, where $i_1, \ldots, i_m \in [n]$ and $z'_1, \ldots, z'_m \in [q]$. Under the event $\mathcal{E}_{\mathsf{good}}$, we have that, for at least $|\mathcal{I}| - 2\eta m \geq m - 2m^2/n - 2\eta m$ values of $j \in [m]$, $z'_j = \pi_{i_j}(c_{\sigma(i_j)} + o_{\sigma(i_j)})$, where we recall that $c \in [q]^n$ is defined on Line 8 of Encode.

Moreover, by Lemma 5.9, for *any* channel $\mathcal{E} \in \mathscr{E}^{\mathsf{H},\mathsf{E}}_{1/2-p_{\mathsf{H}},\varepsilon_{\mathsf{E}}}$, letting $y' = \mathcal{E}(y)$ (so that $y' \in \mathcal{B}_{\mathsf{H},\mathsf{E}}(y, 1/2 - p_{\mathsf{H}}, \varepsilon_{\mathsf{E}})$), there are at least $m \cdot \left(\frac{p_{\mathsf{Sub}}}{2} - \frac{1}{C_{\mathsf{Dec}}} - \frac{2\ell \cdot (nq)^{1-p_{\mathsf{Sub}}^2 + \varepsilon_{\mathsf{Dec}} \log(6e) \log(1/\varepsilon_{\mathsf{Dec}})}}{L_{\mathsf{max}}} \right)$ values of

$j \in [m]$ for which $z'_j \in \mathcal{L}_{i_j}$, where $\mathcal{L}_{i_j}$ denotes the value of the list at the end of $\mathsf{Decode}(1^\lambda, \mathsf{sk}, y')$. Overall, the number of values of $i \in [n]$ for which $i_j = i$ for some $j \in [m]$ and $c_i + o_i \in \mathcal{L}_i$ is at least

$$m \cdot \left(\frac{p_{\mathsf{Sub}}}{2} - \frac{1}{C_{\mathsf{Dec}}} - \frac{2\ell \cdot (nq)^{1-p_{\mathsf{Sub}}^2+\varepsilon_{\mathsf{Dec}} \log(6e) \log(1/\varepsilon_{\mathsf{Dec}})}}{L_{\mathsf{max}}} - \frac{2m}{n} - 2\eta \right). \tag{4}$$

Using the fact that $C_{\mathsf{Dec}} \leq p_{\mathsf{Sub}}/16$, that $2\eta \leq p_{\mathsf{Sub}}/16$, that $2m/n \leq p_{\mathsf{Sub}}/16$ for sufficiently large security parameter λ (as p_{Sub} is a constant), that $q \leq (sn)^s$ (recalling the definition of $q = q(\lambda), n = n(\lambda)$), and that $\varepsilon_{\mathsf{Dec}} \log(6e) \log(1/\varepsilon_{\mathsf{Dec}}) \leq p_{\mathsf{Sub}}^2/2$ (by the choice of $\varepsilon_{\mathsf{Dec}}$) we see that the quantity in (4) is at least

$$m \cdot \left(\frac{p_{\mathsf{Sub}}}{4} - \frac{2\ell \cdot (sn)^{(s+1)\cdot(1-p_{\mathsf{Sub}}^2/2)}}{L_{\mathsf{max}}} \right) \geq m \cdot \left(\frac{p_{\mathsf{Sub}}}{4} - \frac{n^{s-4}}{L_{\mathsf{max}}} \right) \geq \frac{m \cdot p_{\mathsf{Sub}}}{8},$$

where the first inequality uses that $2\ell \cdot (sn)^{(s+1)\cdot(1-p_{\mathsf{Sub}}^2/2)} \leq 2\ell s^{s+1} \cdot n^{s-5} \leq n^{s-4}$ for sufficiently large n (which in turn uses our choice of the constant s), and the second inequality uses that $L_{\mathsf{max}} = n^{s-3} \geq n^{s-4} \cdot 8/p_{\mathsf{Sub}}$ for sufficiently large n by our choice of L_{max}.

Moreover, the lists $\mathcal{L}_1, \ldots, \mathcal{L}_n$ passed to the list recovery algorithm on Line 24 have size bounded by $L_{\mathsf{max}} = n^{s-3}$, which means, by Theorem 5.2, as long as

$$\frac{m \cdot p_{\mathsf{Sub}}}{8} \geq k \cdot \left(n \cdot n^{s-3}\right)^{1/(s+1)} + 2,$$

the list recovery algorithm will return at least one codeword (namely, the codeword chosen by $\mathsf{Encode}(1^\lambda, \mathsf{sk})$ to produce y). In turn, the above inequality holds for sufficiently large n since we have chosen $m = n^{s/(s+1)}$ and $k = n^{1/(s+1)}$. □

Lemma 5.11. *Suppose that $\varepsilon_{\mathsf{E}} \leq 1/400$. Given the parameters of Definition 5.7, the PRC of Algorithm 1 is strongly robust to ε_{E}-edit distance bounded channels.*

Proof. We will show that for any value of sk,

$$\Pr_{y \leftarrow \mathsf{Encode}(1^\lambda, \mathsf{sk})} \left(\forall \mathcal{E} \in \mathscr{O}^{\mathsf{E}}_{\varepsilon_{\mathsf{E}}},\ \mathsf{Decode}(1^\lambda, \mathsf{sk}, \mathcal{E}(y)) \neq \emptyset \right) \geq 1 - \exp\left(-n^{\Omega(1)}\right). \tag{5}$$

Fix any value for sk. Let $\mathcal{E}_{\mathsf{good}}$ be the event that the following events hold:

- Amongst the indices $i_1, \ldots, i_m$ generated on Line 10 of Encode, there are at least $m - 2m^2/n$ distinct elements.
- The substitution channel SC_η on Line 8 of Encode corrupts at most $2\eta m$ of the m symbols of c given by $c_{\sigma(i_1)}, \ldots, c_{\sigma(i_m)}$.

As in the proof of Lemma 5.10, we have $\Pr(\mathcal{E}_{\mathsf{good}}) \geq 1 - \exp\left(-n^{\Omega(1)}\right)$, and we let $\mathcal{I} := \{i_1, \ldots, i_m\} \subset [n]$.

The output of $\mathsf{Encode}(1^\lambda, \mathsf{sk})$, which we denote by $y \in \{0,1\}^{m\ell}$, is the concatenation of $\mathsf{bin}(i_1), \mathsf{bin}(z'_1), \ldots \mathsf{bin}(i_m), \mathsf{bin}(z'_m)$, where $i_1, \ldots, i_m \in [n]$ and $z'_1, \ldots, z'_m \in [q]$. Under the event $\mathcal{E}_{\mathsf{good}}$, we have that, for at least $|\mathcal{I}| - 2\eta m \geq m - 2m^2/n - 2\eta m$ values of $j \in [m]$, $z'_j = \pi_{i_j}(c_{\sigma(i_j)} + o_{\sigma(i_j)})$, where we recall that $c \in [q]^n$ was generated on Line 8 of Encode.

Moreover, by Lemma 5.9, for *any* channel $\mathcal{E} \in \mathscr{C}^{\mathsf{E}}_{\varepsilon_{\mathsf{E}}}$, letting $y' = \mathcal{E}(y)$ (so that $y' \in \mathcal{B}_{\mathsf{E}}(y, \varepsilon_{\mathsf{E}})$), there are at least $m \cdot \left(1 - \frac{1}{C_{\mathsf{Dec}}} - \frac{2\ell \cdot (nq)^{\varepsilon_{\mathsf{Dec}} \log(6e) \log\left(1/\varepsilon_{\mathsf{Dec}}\right)}}{L_{\mathsf{max}}}\right)$ values of $j \in [m]$ for which $z'_j \in \mathcal{L}_{i_j}$, where $\mathcal{L}_{i_j}$ denotes the value of the corresponding list at the end of $\mathsf{Decode}(1^\lambda, \mathsf{sk}, y')$. Overall, the number of values of $i \in [n]$ for which $i_j = i$ for some $j \in [m]$ and $c_i + o_i \in \mathcal{L}_i$ is at least

$$m \cdot \left(1 - \frac{1}{C_{\mathsf{Dec}}} - \frac{2\ell \cdot (nq)^{\varepsilon_{\mathsf{Dec}} \log(6e) \log(1/\varepsilon_{\mathsf{Dec}})}}{L_{\mathsf{max}}} - \frac{2m}{n} - 2\eta\right). \tag{6}$$

Using the fact that $C_{\mathsf{Dec}} \leq 1/16$, that $2\eta \leq 1/16$, that $2m/n \leq p_{\mathsf{Sub}}/16$ for sufficiently large security parameter λ (as p_{Sub} is a constant), that $q \leq n$ (recalling the definition of $q = q(\lambda), n = n(\lambda)$), and that $\varepsilon_{\mathsf{Dec}} \log(6e) \log(1/\varepsilon_{\mathsf{Dec}}) \leq 1/10$ (by the choice of $\varepsilon_{\mathsf{Dec}} = 2\varepsilon_{\mathsf{E}} C_{\mathsf{Dec}} = 16\varepsilon_{\mathsf{E}}$ and $\varepsilon_{\mathsf{E}} \leq 1/400$) we see that the quantity in (6) is at least

$$m \cdot \left(\frac{1}{2} - \frac{2\ell \cdot n^{1/5}}{L_{\mathsf{max}}}\right) \geq \frac{m}{4},$$

where we have used that $L_{\mathsf{max}} = n^{2/5} \geq 2\ell \cdot n^{1/5}$ for sufficiently lage n.

Moreover, the lists $\mathcal{L}_1, \ldots, \mathcal{L}_n$ passed to the list recovery algorithm on Line 24 have size bounded by $L_{\mathsf{max}} = n^{2/5}$, which means, by Theorem 5.1, as long as

$$\frac{m}{4} \geq \sqrt{k n^{2/5} n} = n^{4/5},$$

the list recovery algorithm will return at least one codeword (namely, the codeword chosen by $\mathsf{Encode}(1^\lambda, \mathsf{sk})$ to produce y). In turn, the above inequality holds by our definition of m. □

Lemma 5.12. *The PRC of Algorithm 1 (with the parameter settings of either Definition 5.7 or Definition 5.8) is sound.*

Proof. Fix any string $y \in \{0,1\}^\star$. Note that the lists $\mathcal{L}_1, \ldots, \mathcal{L}_n$ as constructed in $\mathsf{Decode}(1^\lambda, \mathsf{sk}, y)$ do not depend on o. Recall that we use $C = C(\lambda)$ to denote the code used in Algorithm 1. Thus, it suffices to show that, for any sets $\mathcal{L}_1, \ldots, \mathcal{L}_n$, each of size at most L_{max}, we have

$$\Pr_{o \sim \mathrm{Unif}([q]^n)} \left(\exists y^\star \in C \text{ s.t. } y^\star_w + o_w \in \mathcal{L}_w \text{ for at least } t_{\mathsf{rec}} \text{ values of } w \in [n]\right) \leq \exp\left(n^{-\Omega(1)}\right). \tag{7}$$

In turn, we may verify (7) by a simple counting argument. The number of strings $y \in [q]^n$ which satisfy $y_i \in \mathcal{L}_i$ for at least t_{rec} values of $i \in [n]$ is bounded above by

$$q^{n - t_{\mathsf{rec}}} \cdot L_{\mathsf{max}}^{t_{\mathsf{rec}}} \cdot \binom{n}{t_{\mathsf{rec}}} \leq q^{n - t_{\mathsf{rec}}} \cdot \left(\frac{enL_{\mathsf{max}}}{t_{\mathsf{rec}}}\right)^{t_{\mathsf{rec}}} = q^n \cdot \left(\frac{enL_{\mathsf{max}}}{qt_{\mathsf{rec}}}\right)^{t_{\mathsf{rec}}}.$$

Then (7) follows using the fact that $o \sim \mathrm{Unif}([q]^n)$ together with a union bound over the q^k choices of $y^\star \in C$, as we work out for each of the two parameter settings below:

- For the parameter settings in Definition 5.7, we have $t_{\mathsf{rec}} = \sqrt{kL_{\mathsf{max}} n} = n^{4/5}$, and so

$$q^k \cdot \left(\frac{enL_{\mathsf{max}}}{qt_{\mathsf{rec}}}\right)^{t_{\mathsf{rec}}} \leq n^{n^{1/5}} \cdot \left(\frac{2en \cdot n^{2/5}}{n \cdot n^{4/5}}\right)^{n^{4/5}} \leq \exp\left(-n^{\Omega(1)}\right).$$

- For the parameter settings in Definition 5.8, we have $t_{\mathsf{rec}} = n^{(s-1)/(s+1)}+2$ and $n^s \le q \le (sn)^s$, so

$$q^k \cdot \left(\frac{enL_{\mathsf{max}}}{qt_{\mathsf{rec}}}\right)^{t_{\mathsf{rec}}} \le (sn)^{s\cdot n^{1/(s+1)}} \cdot \left(\frac{en\cdot n^{s-3}}{n^s}\right)^{n^{(s-1)/(s+1)}} \le \exp\left(-n^{\Omega(1)}\right).$$

□

Lemma 5.13. *The PRC of Algorithm 1 is undetectable under the following assumptions:*

- *For the parameter settings in Definition 5.7, under Conjecture 3.2;*
- *For the parameter settings in Definition 5.8, under Conjecture 5.3.*

Proof. Consider any algorithm Adv running in time $T = T(\lambda)$ and which satisfies

$$\left|\Pr_{\mathsf{sk}\leftarrow\mathsf{KeyGen}(1^\lambda)}\left(\mathsf{Adv}^{\mathsf{Encode}(1^\lambda,\mathsf{sk})}(1^\lambda)=1\right) - \Pr_{\mathcal{U}}\left(\mathsf{Adv}^{\mathcal{U}}(1^\lambda)=1\right)\right| = \nu(\lambda),$$

for some functions $\nu, T : \mathbb{N} \to \mathbb{R}_{\ge 0}$. Here $\mathcal{U}$ denotes the oracle that returns a uniformly random string of the same length as $\mathsf{Encode}(1^\lambda, \mathsf{sk})$. We will use Adv to construct an adversary Adv' which distinguishes between the distributions $\mathcal{D}_{n,[q],C,\eta,T}$ and $\mathrm{Unif}(([q]^n)^T)$; here the parameters n, q, m and the code C are given by either Definition 5.7 or Definition 5.8. Note that in either case, the definitions of our parameters ensure that the conjecture in question (i.e., Conjecture 3.2 or Conjecture 5.3) is exactly that the distributions $\mathcal{D}_{n,[q],C,\eta,T}$ and $\mathrm{Unif}(([q]^n)^T)$ are computationally indistinguishable. (We omit the dependence of the various parameters on λ.)

Note that Adv requires oracle access to an oracle $\mathcal{O}$ (to be interpreted as either $\mathsf{Encode}(1^\lambda, \mathsf{sk})$ or the uniform distribution) which when called outputs a sample in $\{0,1\}^{m\cdot(\log q+\log n)}$. The algorithm Adv' simulates Adv as follows: first, Adv' draws $o \sim \mathrm{Unif}([q]^n)$. Then, for $t \in [T]$, for the tth call that Adv makes to the oracle $\mathcal{O}$, Adv' uses the tth sample from its distribution (either $\mathcal{D}_{n,[q],C,\eta,T}$ or $\mathrm{Unif}(([q]^n)^T)$) which is a codeword $c \in [q]^n$, to simulate the oracle call $\mathcal{O}$, as follows:

- It draws $i_1, \dots, i_m \sim \mathrm{Unif}([n])$ independently.
- For each $j \in [m]$, if there is $j' < j$ with $i_j = i_{j'}$, then Adv' sets $z'_j \leftarrow \mathrm{Unif}([q])$. Otherwise, it sets $z'_j \leftarrow c_{i_j} + o_{i_j}$.
- Adv' then uses $\mathsf{bin}(i_1) \circ \mathsf{bin}(z'_1) \circ \cdots \circ \mathsf{bin}(i_m) \circ \mathsf{bin}(z'_m)$ as the output of the oracle $\mathcal{O}$.

It is clear that the running time of Adv' is at most $T(\lambda) \cdot \mathrm{poly}(\lambda)$. We now make the following observations:

- If the distribution that Adv' is given is $\mathcal{D}_{n,[q],C,\eta,T}$, then Adv' simulates $\mathsf{Adv}^{\mathsf{Encode}(1^\lambda,\mathsf{sk})}(1^\lambda)$, where $\mathsf{sk} = (\sigma, \pi_1, \dots, \pi_n, \sigma \circ o)$,[14] with $\sigma, \pi_1, \dots, \pi_n$ being given by the (secret) sampled permutations in $\mathcal{D}_{n,[q],C,\eta,T}$.

  Indeed, in the simulation of the oracle $\mathcal{O} = \mathsf{Encode}(1^\lambda, \mathsf{sk})$ as above, the codeword c (representing one of the T samples of $\mathcal{D}_{n,[q],C,\eta,T}$) is given by $c \leftarrow \mathrm{SC}_\eta(\hat{c})$ where $\hat{c}_i = \pi_i(\tilde{c}_{\sigma(i)})$, where $\tilde{c} \leftarrow C$ is a uniformly random codeword in C. Then the result of the sequence of operations

[14] Here $\sigma \circ o$ denotes $(o_{\sigma(1)}, \dots, o_{\sigma(n)})$.

$\tilde{c} \mapsto \hat{c} \mapsto c \mapsto (i_1, z'_1, \ldots, i'_m, z'_m)$ in the simulation of Adv' has the same distribution as the sequence of operations $c \mapsto c' \mapsto z \mapsto (i_1, z'_1, \ldots, i_m, z'_m)$ in the execution of Algorithm 1 (where $c \leftarrow \mathrm{Unif}(C)$, $c' \leftarrow \mathrm{SC}_\eta(c)$, $z \leftarrow c' + o$, and $z'_j \leftarrow \pi_{i_j}(z_{\sigma(i_j)})$).

- If the distribution that Adv' is given is $\mathrm{Unif}(([q]^n)^T)$, then Adv' simulates $\mathsf{Adv}^{\mathcal{U}}(1^\lambda)$.

  Indeed, in the simulation of the oracle $\mathcal{O} = \mathcal{U}$ as above, the codeword c is uniformly random, which means that so is the sequence $(i_1, z'_1, \ldots, i_m, z'_m)$.

Thus, under the assumption that there is no subexponential-time algorithm distinguishing $\mathcal{D}_{n,[q],C,\eta,T}$ and $\mathrm{Unif}(([q]^n)^T)$, we must have that either $T(\lambda) \geq \exp\left(\lambda^{\Omega(1)}\right)$ or $\nu(\lambda) \leq \exp\left(-\lambda^{\Omega(1)}\right)$. □

5.5 From PRCs to watermarking: background

In this section, we give some background on watermarking schemes for language models; we refer the reader to [CG24] for further details and explanations.

Definition 5.9. An *autoregressive language model* Model over alphabet (i.e., token set) Σ is a deterministic algorithm that takes as input a prompt $\mathtt{PROMPT} \in \Sigma^\star$ and a sequence of previous tokens $\mathsf{t}_1, \ldots, \mathsf{t}_{i-1}$ and outputs a distribution $p_i := \mathsf{Model}(\mathtt{PROMPT}, \mathsf{t}_{1:i-1}) \in \Delta(\Sigma)$.

In the event that $\Sigma = \{0, 1\}$, we abuse notation slightly and write $p_i = \mathsf{Model}(\mathtt{PROMPT}, \mathsf{t}_{1:i-1}) \in [0, 1]$ to denote the probability that the model places on the next token being 1.

We assume that $\mathtt{PROMPT}$ encodes the length of desired text output by Model. Given a model Model, a prompt $\mathtt{PROMPT}$ which specified some length ℓ, we let $\overline{\mathsf{Model}}(\mathtt{PROMPT})$ denote the random variable $\mathsf{t} \in \Sigma^\ell$ generated in the natural way, i.e., given $\mathsf{t}_{1:i-1}$, we draw $\mathsf{t}_i \sim \mathsf{Model}(\mathtt{PROMPT}, \mathsf{t}_{1:i-1})$, and repeat for ℓ steps.

In order to obtain watermarking procedures for language models, we need to assume that the model has sufficient *entropy*, formalized below:

Definition 5.10. Given a language model Model, a sequence $\mathsf{t} \in \Sigma^\star$, and $i, j \in [|\mathsf{t}|]$, we define the *empirical entropy* of Model on t on the subsequence $[i, j]$ by

$$H_{\mathsf{e}}^{[i:j]}(\mathsf{t}, \mathsf{Model}) := \sum_{a=i}^{j} -\log \mathsf{Model}(\mathsf{t}_a \mid \mathsf{t}_{1:a-1}).$$

When the choice of Model is clear from context, we will often write $H_{\mathsf{e}}^{[i:j]}(\mathsf{t}) := H_{\mathsf{e}}^{[i:j]}(\mathsf{t}, \mathsf{Model})$. Moreover, we write $H_{\mathsf{e}}^{[i:j)}(\mathsf{t}) = H_{\mathsf{e}}^{[i:j-1]}(\mathsf{t}, \mathsf{Model})$.

A *watermarking scheme* $\mathcal{W}$ for a model Model is a tuple of polynomial-time algorithms $\mathcal{W} = (\mathsf{Setup}, \mathsf{Wat}, \mathsf{Detect})$, where:

- $\mathsf{Setup}(1^\lambda)$ outputs a secret key sk of length polynomial in λ, where $\lambda \in \mathbb{N}$ denotes a security parameter.
- $\mathsf{Wat}(\mathsf{sk}, \mathtt{PROMPT})$ takes as input a prompt and sk and outputs a response $\mathsf{t} \in \Sigma^\ell$ of length ℓ.
- $\mathsf{Detect}(\mathsf{sk}, \mathsf{t})$ takes as inptut sk and a sequence $\mathsf{t} \in \Sigma^\star$ and outputs a response in $\{\mathsf{True}, \mathsf{False}\}$ indicating whether Detect detects the sequence t as being watermarked according to sk.

The main desired properties of watermarking schemes parallel those of PRCs, and are formalized below:

Definition 5.11 (Undetectability ([CGZ24]))**.** A watermarking scheme $\mathcal{W}$ is *undetectable* if for every security parameter λ, prompt $\texttt{PROMPT}$, and any polynomial-time distinguisher Dist, it holds that

$$\left|\Pr\left(\mathsf{Dist}^{\overline{\mathsf{Model}}}(1^\lambda) = \mathsf{True}\right) - \Pr\left(\mathsf{Dist}^{\mathsf{Wat}(\mathsf{sk},\cdot)}(1^\lambda) = \mathsf{True}\right)\right| \leq \mathsf{negl}(\lambda),$$

where $\mathsf{Dist}^{\mathcal{O}}$ means that the distinguisher can query $\mathcal{O}$ with an arbitrary prompt $\texttt{PROMPT}$.

Definition 5.12 (Soundness)**.** A watermarking scheme $\mathcal{W}$ is *sound* if for every security parameter λ and every fixed token sequence $\mathsf{t} \in \Sigma^\star$ of length $\mathrm{poly}(\lambda)$,

$$\Pr_{\mathsf{sk}\leftarrow\mathsf{Setup}(1^\lambda)}(\mathsf{Detect}(\mathsf{sk},\mathsf{t}) = \mathsf{True}) \leq \mathsf{negl}(\lambda).$$

Definition 5.13 (Substring Robustness)**.** Fix a channel $\mathcal{E} : \{0,1\}^\star \times \Sigma^\star \to \Sigma^\star$ that takes as input a codeword $x \in \Sigma^\star$ and some auxiliary information $\mathsf{sk} \in \{0,1\}^\star$. Consider a family of functions $\beta_\lambda : \mathbb{N} \to \mathbb{N}$ indexed by the security parameter λ. Then a watermarking scheme $\mathcal{W}$ is defined to be *β-robust to $\mathcal{E}$* if for each $\lambda \in \mathbb{N}$ and each prompt $\texttt{PROMPT}$,

$$\Pr_{\substack{\mathsf{sk}\leftarrow\mathsf{Setup}(1^\lambda)\\ \mathsf{t}\leftarrow\mathsf{Wat}(\mathsf{sk},\texttt{PROMPT}),\ \mathsf{t}'\leftarrow\mathcal{E}(\mathsf{sk},\mathsf{t})}}\left(\exists i,j \in [|\mathsf{t}|],\ \text{ s.t. } \mathsf{Detect}(\mathsf{sk},\mathsf{t}') = \mathsf{False} \text{ and } H_{\mathsf{e}}^{[i:j]}(\mathsf{t}) \geq \beta_\lambda(j-i)\right) \leq \mathsf{negl}(\lambda).$$

5.6 From PRC to watermarking: soundness, undetectability, & robustness

In this section, we establish the following theorem, which shows that we can obtain watermarking schemes for language models which have strong adaptive robustness to edit-bounded channels whenever the per-token entropy of the language model is at least a constant. For simplicity, we focus on binary-alphabet language models, but a straightforward reduction (see [CG24]) shows that this is without loss of generality.

Theorem 5.14 (Main edit-robust watermarking result)**.** *Fix any constant $\alpha > 0$ and write, for security parameter λ, $\beta_\lambda(\ell) = 8\alpha \cdot \ell + 2\sqrt{2} \cdot \lambda$. Then there is a watermarking scheme for any language model over a binary alphabet which is β-robust to any $\tilde{O}(\alpha^7)$-edit-bounded channel and which satisfies soundness and undetectability, where the latter holds under Conjecture 5.3. The key generation and watermarking procedures run in time $\mathrm{poly}(\lambda)$, and the detection procedure runs in time $\lambda^{O(1/\alpha^4)}$.*

Intuitively, the guarantee of β-robustness to $\tilde{O}(\alpha^7)$-edit-bounded channels in the context of Theorem 5.14 means the following: as long as the empirical entropy of the language model for a sequence of length $\ell \gg \lambda$ is at least an $\Omega(\alpha)$-fraction of the length, then we can still detect the watermark even after the sequence has been passed through an arbitrary channel which can make a $\tilde{O}(\alpha^7)$-fraction of edits. Improving the exponent in the bound on the fraction of edits and improving the running time of the detection procedure are both interesting open questions.

The watermarking procedure of Theorem 5.14 is presented in Algorithm 2. It depends on a PRC PRC; given such a PRC, we let $\mathcal{W}[\mathsf{PRC}]$ denote the watermarking scheme of Algorithm 2. It

Algorithm 2 Binary-alphabet watermarking scheme from binary-alphabet PRC [CG24]

Require: Pseudorandom code $\mathsf{PRC} = (\mathsf{PRC.KeyGen}, \mathsf{PRC.Encode}, \mathsf{PRC.Decode})$. Maximum length I_{max} for the watermarked text as encoded by any possible prompt.

function $\mathsf{Setup}(1^\lambda)$
  $\mathsf{PRC.sk} \leftarrow \mathsf{PRC.KeyGen}(1^\lambda)$.
  $\sigma_1, \ldots, \sigma_{I_{\mathsf{max}}} \leftarrow \mathrm{Unif}(\{0,1\}^{n(\lambda)})$.
  return $\mathsf{sk} := (\mathsf{PRC.sk}, \sigma_{1:I_{\mathsf{max}}})$.
function $\mathsf{Wat}(\mathsf{sk}, \texttt{PROMPT})$
  Let $\ell \in \mathbb{N}$ denote the desired length encoded by `PROMPT` and $n = n(\lambda)$ denote the block length of the PRC corresponding to $\mathsf{PRC.sk}$.
  Write $\mathsf{sk} = (\mathsf{PRC.sk}, \sigma_{1:I_{\mathsf{max}}})$.
  for $1 \le i \le \ell$ **do**
    if $i \equiv 1 \pmod{n}$ **then**
      Set $x \leftarrow \sigma_{\lceil i/n \rceil} \oplus \mathsf{PRC.Encode}(\mathsf{sk}) \in \{0,1\}^n$.
    $p_i \leftarrow \mathsf{Model}(\texttt{PROMPT}, \mathsf{t}_{1:i-1})$.
    Draw $\mathsf{t}_i \leftarrow \mathrm{Ber}(p_i - (-1)^{x_i} \cdot \min\{p_i, 1-p_i\})$.
function $\mathsf{Detect}(\mathsf{sk}, \mathsf{t})$
  for $i \in [|\mathsf{t}|], j \in [i, \min\{i+n, |\mathsf{t}|]$ **do**
    if $\mathsf{PRC.Decode}(\mathsf{sk}, \mathsf{t}_{i:j-1}) = \mathsf{True}$ **then**
      return True
  return False

also takes as input a parameter I_{max} denoting the maximum possible length of a sequence output by the model, as encoded by any possible prompt; we assume that I_{max} is polynomial in the security parameter. The construction is essentially identical to that of [CG24]; moreover, [CG24] showed that the properties of soundness and undetectability of $\mathcal{W}[\mathsf{PRC}]$ follow from those of PRC, and these guarantees carry over in a straightforward manner to our setting:

Lemma 5.15 ([CG24], Lemma 18)**.** *Suppose that* PRC *is sound. Then* $\mathcal{W}[\mathsf{PRC}]$ *is sound.*

Lemma 5.16 ([CG24], Lemma 19)**.** *Suppose that* PRC *is undetectable. Then* $\mathcal{W}[\mathsf{PRC}]$ *is undetectable.*

The next lemma shows that if PRC has strong adaptive robustness to the set of substitution-edit bounded channels (for a sufficiently large substitution error rate), then $\mathcal{W}[\mathsf{PRC}]$ has strong adaptive robustness to the set of edit-bounded channels.

Lemma 5.17. *Let* $\varepsilon_{\mathsf{H}}, \varepsilon_{\mathsf{E}} > 0$ *be constants, and suppose that* $\mathscr{E}$ *is the set of channels* $\mathcal{E} : \Sigma^\star \to \Sigma^\star$ *which are* $(1/2 - \varepsilon_{\mathsf{H}}, \varepsilon_{\mathsf{E}})$*-substitution-edit bounded. If* PRC *is robust to every channel in* $\mathscr{E}$*, then for* $\beta_\lambda(\ell) := 8\sqrt{\varepsilon_{\mathsf{H}}} \cdot \ell + 2\sqrt{2} \cdot n(\lambda)$*, we have that* $\mathcal{W}[\mathsf{PRC}]$ *is* β*-robust to any* $\varepsilon_{\mathsf{E}}\sqrt{\varepsilon_{\mathsf{H}}}/3$*-edit-bounded channel.*

Proof. Fix a prompt $\mathtt{PROMPT}$, and let $\mathsf{t} \leftarrow \mathsf{Wat}(\mathsf{sk}, \mathtt{PROMPT})$ denote the output of the watermarking procedure given $\mathtt{PROMPT}$. Write $\varepsilon'_{\mathsf{E}} := \varepsilon_{\mathsf{E}}\sqrt{\varepsilon_{\mathsf{H}}}/3$, and fix any $\varepsilon'_{\mathsf{E}}$-edit bounded channel $\mathcal{E}$, and let $\mathsf{t}' = \mathcal{E}(\mathsf{t})$. Let us write $\ell := |\mathsf{t}|$, and fix any $i, j \in [\ell]$ with $j \geq i$. Let i' denote the smallest value which is 1 (mod n) and at least i and let j' denote the largest value which is 1 (mod n) and at most j. Let $i_1 = i', i_2, \ldots, i_{g+1} = j'$ denote the values which are 1 (mod n) and between i' and j'. Moreover let $x_1, \ldots, x_g \in \{0,1\}^n$ denote the PRC codewords which are generated on Line 10 of Algorithm 2 corresponding to positions $i_1, \ldots, i_{g+1}$ (after the addition with the one-time pads σ_i). For each $f \in [g]$, we consider the following *embedding channel* $\mathcal{E}_f^{\mathsf{Emb}} : \{0,1\}^n \to \{0,1\}^\star$. $\mathcal{E}_a^{\mathsf{Emb}}$ maps the string $x_f \in \{0,1\}^n$ to the substring $\mathsf{t}_{i_f:i_{f+1}-1}$ of Model's output. Note that $\mathcal{E}_f^{\mathsf{Emb}}$ is randomized, as it depends on the other PRC codewords as well as the randomness of the draws of t_i on Line 12 of Algorithm 2.

By [CG24, Lemma 20], for any $k, k' \in [\ell]$ with $k' - k = n$, with probability $1 - \mathsf{negl}(n)$, we have $H_{\mathsf{e}}^{[k,k')}(\mathsf{t}) \leq \sqrt{2} \cdot n$. It follows in particular that with probability $1 - \mathsf{negl}(n)$, $H_{\mathsf{e}}^{[i,i')}(\mathsf{t}) \leq \sqrt{2} \cdot n$, and $H_{\mathsf{e}}^{[j',j)}(\mathsf{t}) \leq \sqrt{2} \cdot n$. Thus, with probability $1 - \mathsf{negl}(n)$, $H_{\mathsf{e}}^{[i',j')}(\mathsf{t}) \geq 8\sqrt{\varepsilon_{\mathsf{H}}} \cdot \ell \geq 8\sqrt{\varepsilon_{\mathsf{H}}} \cdot ng$. Hence there are at least $\sqrt{\varepsilon_{\mathsf{H}}} \cdot g$ values of $f \in [g]$ for which $H_{\mathsf{e}}^{[i_f,i_{f+1})}(\mathsf{t}) > 4\sqrt{\varepsilon_{\mathsf{H}}} \cdot n$. Let the set of such f be denoted by $\mathcal{S} \subset [g]$.

For each $f \in [g]$, let e_f denote the number of edits (i.e., insertions and deletions) made at positions $[i_f, i_{f+1})$ to obtain t' from t (via $\mathcal{E}$). Note that at least one of these values of $f \in \mathcal{S}$ must have the property that $e_f \leq n \cdot 3\varepsilon'_{\mathsf{E}}/\sqrt{\varepsilon_{\mathsf{H}}} = n \cdot \varepsilon_{\mathsf{E}}$ (as otherwise the total number of edits made to obtain t' from t would be greater than $3ng \cdot \varepsilon'_{\mathsf{E}} \geq \ell \cdot \varepsilon'_{\mathsf{E}}$). Let $f^\star \in [g]$ denote the random variable which is the smallest such value of f. Due to the addition of the one-time pads σ_f to each x_f, each of the codewords x_f is drawn uniformly from $\{0,1\}^n$, independently from other $x_{f'}$ and also from the channel $\mathcal{E}_f^{\mathsf{Emb}}$. We now need the following result, which is a direct consequence of Lemma 21 of [CG24] together with the fact we have just stated that x_f is uniform and independent of $\mathcal{E}_f^{\mathsf{Emb}}$ (in particular, we use here that the prompt $\mathtt{PROMPT}$ and therefore the model's distribution do not depend on sk):

Lemma 5.18 (Lemma 21 of [CG24]). *For any constant $c > 0$ and value of $f \in [g]$, it holds that*

$$\Pr_{\substack{\mathsf{sk}\leftarrow\mathsf{KeyGen}(1^\lambda)\\ x\leftarrow\mathsf{PRC.Encode}(\mathsf{sk})\\ x'\leftarrow\mathcal{E}_f^{\mathsf{Emb}}(x)}}\left(D_{\mathsf{H}}(x,x') > \left(\frac{1}{2}-\frac{c^2}{16}\right)\cdot n \text{ and } H_{\mathsf{e}}^{[i_f:i_{f+1})}(x') > c\cdot n\right) \le \mathsf{negl}(n).$$

It follows from Lemma 5.18 (with $c = 4\sqrt{\varepsilon_{\mathsf{H}}}$) that with probability $1 - \mathsf{negl}(n)$, we have that $D_{\mathsf{H}}(x_{f^\star}, \mathsf{t}_{i_{f^\star}:i_{f^\star+1}-1}) \le \left(\frac{1}{2} - \varepsilon_{\mathsf{H}}\right)\cdot n$. Moreover, by choice of $f^\star$, we also have $D_{\mathsf{E}}(\mathsf{t}_{i_{f^\star}:i_{f^\star+1}-1}, \mathsf{t}'_{a:b}) \le \varepsilon_{\mathsf{E}}\cdot n$. Thus, since PRC is strongly robust to every channel which is $(1/2-\varepsilon_{\mathsf{H}}, \varepsilon_{\mathsf{E}})$-substitution-edit bounded, with probability $1 - \mathsf{negl}(n)$, we have that $\mathsf{PRC.Decode}(\mathsf{sk}, \mathsf{t}'_{a:b-1}) = \mathsf{True}$ for some $a, b \in [\ell]$. □

Finally we are ready to prove Theorem 5.14.

Proof of Theorem 5.14. We use the PRC of Algorithm 1 with the parameter settings of Definition 5.8, so that Theorem 5.7 applies. The soundness and undetectability guarantees of Theorem 5.7, together with Lemma 5.15 and Lemma 5.16, imply that $\mathcal{W}[\mathsf{PRC}]$ satisfies soundness and undetectability (under Conjecture 5.3). Finally, taking $\beta_\lambda(\ell) = 8\alpha\cdot\ell + 2\sqrt{2}\cdot n(\lambda)$, by applying Lemma 5.17 with $\varepsilon_{\mathsf{H}} = \alpha^2$ and Theorem 5.7 with $p_{\mathsf{Sub}} = \varepsilon_{\mathsf{H}} = \alpha^2$ (which requires us to take $\varepsilon_{\mathsf{E}} \le \tilde{O}(p_{\mathsf{Sub}}^3) = \tilde{O}(\alpha^6)$), we obtain that $\mathcal{W}[\mathsf{PRC}]$ is β-robust to any $\tilde{O}(\alpha^7)$-edit-bounded channel. The running time guarantees follow directly from those of Algorithm 1. □

6 Substitution-robust pseudorandom codes

In this section, we give a straightforward construction which converts any family of codes satisfying the permuted codes assumption Definition 3.1 into a PRC that has strong adaptive robustness to any substitution channel introducing at most a constant fraction of errors. (Under Conjecture 3.1, this construction works for any code family with large dual distance.) Of course, such a result follows from the construction based on Reed-Solomon codes in Section 5, under the permuted codes assumption for Reed-Solomon codes (Conjecture 3.2). The construction in this section, however, is secure as long as the conjecture holds for *some* family of codes with high dual distance and efficient decoding from a constant error rate.

Proposition 6.1. *Let $\lambda \in \mathbb{Z}^+$ be a security parameter, let $n = n(\lambda), k = k(\lambda), q = q(\lambda)$ be polynomially-bounded functions in λ denoting the block length, dimension, and alphabet size, respectively. Let $C = \{C(\lambda)\}_{\lambda\in\mathbb{N}}$ be a family of linear codes with $C(\lambda) \subset \mathbb{F}_{q(\lambda)}^{n(\lambda)}$. Suppose that C comes equipped with efficient algorithms $\mathsf{Encode}_C : \mathbb{F}_q^k \to \mathbb{F}_q^n$ and $\mathsf{Decode}_C : \mathbb{F}_q^n \to \mathbb{F}_q^k \cup \{\perp\}$ which correct up to a fraction $\delta \le 1 - 1/q - \Omega(1)$ of worst-case substitutions. Under the permuted codes assumption for C with error $\eta = \delta/3$ (Definition 3.1), the PRC of Algorithm 3 for the family C is undetectable, sound, and has strong adaptive robustness to any channel $\mathcal{E}$ introducing a fraction of at most $\delta/2$ of substitutions.*

We remark that the robustness of the PRC in Proposition 6.1 can be improved to essentially the same rate δ as is achieved by the underlying code C, decreasing the error rate η to a vanishing fraction of δ. Moreover, it is possible to further improve the robustness by using a code C which has an efficient *list-decoding* algorithm.

Below we outline some concrete families of code C which have polynomial dual distance, and thus, under Conjecture 3.1, satisfy the hypotheses of Proposition 6.1. We emphasize that the below families all have alphabet size $q(\lambda)$ which is a *constant* (i.e., does not depend on λ).

Example: AG codes. We follow the terminology and notation of [HvL98]. Let q be the square of a prime; then [SG95] shows the existence of, for each $m \in \mathbb{N}$, an irreducible nonsingular projective curve $\mathcal{X}$ over $\mathbb{F}_q$ with genus less than $g := q^{(m-1)/2} \cdot (\sqrt{q}+1)$ and with more than $n := q^{(m-1)/2} \cdot (q-1)$ rational points. Letting $n+1$ of these rational points be denoted $Q, P_1, \ldots, P_n \in \mathcal{X}$, we define the divisors $D := P_1 + \cdots + P_n$ and $G := a \cdot Q$ for some integer $a > 2g-2$. Let $\mathcal{L}(G) := \{f \in \mathbb{F}(\mathcal{X})^\star \mid (f)+G \geq 0\} \cup \{0\}$ denote the vector space of functions on $\mathcal{X}$ with poles only at Q of order at most a. We consider the AG code $C = C(D,G)$, defined as the image of the linear mapping $E : \mathcal{L}(G) \to \mathbb{F}_q^n$ defined by $E(f) = (f(P_1), \ldots, f(P_n))$. [HvL98, Theorems 2.65 & 2.69] give the following bounds on the dimension k, distance d, and dual distance $d^\star$ of $C(D,G)$:

$$k = a - g + 1, \qquad d \geq n - a, \qquad d^\star \geq a - 2g + 2.$$

Moreover, there are efficient algorithms to decode $C(D,G)$ up to $\lfloor (d-1)/2 \rfloor$ substitutions [HvL98, Theorem 6.12].[15] Taking q to be a constant (e.g., any prime square at least 25 will suffice), for each $m \in \mathbb{N}$, we take the degree of the divisor Q to be $a_m := 2g_m - 2$, and we obtain a code $C = C_m$ with block length $n_m = q^{(m-1)/2} \cdot (q-1) = g_m \cdot (\sqrt{q}-1)$, whose distance d_m and dual distance $d_m^\star$ satisfy

$$d_m^\star = a_m - 2g_m + 2 = g_m = \frac{n_m}{\sqrt{q}-1} = \Omega(n_m), \qquad d_m \geq n_m - a_m = g_m \cdot (\sqrt{q} - 1 - 3) = \Omega(n_m),$$

i.e., both are a constant fraction of the block length. Thus, under Conjecture 3.1, the resulting family of codes satisfies the assumptions of Proposition 6.1. Notice also that we can get a similar result (namely, linear distance and dual distance) over a binary alphabet by concatenating with the trivial code, though the distance and dual distance will of course degrade by a constant factor.

Example: Raw Reed-Solomon Codes. As another example, we consider the *Raw Reed-Solomon* codes introduced in [SKS19]. For a positive integer m, we let $r = 2^m$, we write $\tilde{n} := r - 1$, and fix some $\tilde{k} < \tilde{n}$ denoting the dimension (which will ultimately be taken to be $k = \tilde{n}^\alpha$ for some $\alpha \in (0, 1/2)$). We first consider the Reed-Solomon code $\tilde{C} := \mathsf{RS}_{\mathbb{F}_r, \tilde{n}, \tilde{k}}$ over $\mathbb{F}_r$ with evaluation points $\mathbb{F}_r^\star$ (so that the block length is $\tilde{n} = r - 1$). Interpreting $\mathbb{F}_r$ as a dimension-m vector space over $\mathbb{F}_2$ via a bijection $\Phi : \mathbb{F}_r \simeq \mathbb{F}_2^m$ allows us to interpret $\tilde{C}$ as a code $C \subset \mathbb{F}_2^{\tilde{n}m}$ with dimension $k := \tilde{k}m$. While the distance of $\tilde{C}$ is only guaranteed to be at least $\tilde{n} - \tilde{k}$, which is *sublinear* in the block length $\tilde{n}m$ as $m = \Theta(\log \tilde{n})$, a remarkable fact shown in [SKS19] states that if we consider the subcode of $\tilde{C}$ induced by only encoding polynomials with constant term equal to 0, then the resulting binary distance has linear distance; in fact, its relative distance approaches $1/2$. In particular, as shown in [SKS19, Theorem 3.1], this construction gives, for each m, and $\alpha \in (0, 1/2)$, a code C with block length $n_m = (2^m - 1) \cdot m$, dimension $k_m = n_m^\alpha$, and whose distance d_m and dual

[15] See also, e.g., [SW98, Corollary 5.2] for a particularly simply algorithm which decodes up to $\lfloor (d-1)/2 \rfloor - g$ substitutions. Additionally, we remark that all of these algorithms assume the existence of a succinctly-represented basis for $\mathcal{L}(G)$; we assume that such a basis exists. See e.g., [Hes02].

distance $d_m^\star$ satisfy

$$d_m^\star = \Omega\left(\frac{n_m^\alpha}{\log n_m}\right), \qquad d_m = n_m \cdot \left(\frac{1}{2} - O\left(\left(\frac{\log n}{n}\right)^{1/2-\alpha}\right)\right) = \left(\frac{1}{2} - o(1)\right) \cdot n_m.$$

Moreover, this code is efficiently decodable up to half its distance; thus, under Conjecture 3.1, the resulting family of codes satisfies the assumptions of Proposition 6.1.

6.1 Proof of Proposition 6.1

The proof of Proposition 6.1 is straightforward and follows similar (but simpler) lines to those in Section 5: in particular, we prove undetectabilty, soundness, and robustness in the below lemmas:

Lemma 6.2. *In the setting of Proposition 6.1, the PRC of Algorithm 3 is sound.*

Proof. Fix any string $y \in \{0,1\}^\star$. It suffices to show that

$$\Pr_{o \sim \mathrm{Unif}(\mathbb{F}_q^n)}(\exists y^\star \in C \text{ s.t. } D_{\mathsf{H}}(y^\star, y - o) \le \delta \cdot n) \le \exp(-\Omega(n)). \tag{8}$$

The number of strings $y' \in \mathbb{F}_q^n$ which satisfy $D_{\mathsf{H}}(y', y-o) \le \delta \cdot n$ is at most $q^{H_q(\delta)\cdot n}$, where $H_q(\cdot)$ denotes the q-ary entropy function. Since we have assumed $\delta \le 1 - 1/q - \Omega(1)$, then this number is at most $q^{(1-\Omega(1))\cdot n}$, meaning that the probability of the event in (8) is at most $q^{-\Omega(n)}$, as desired. □

Lemma 6.3. *In the setting of Proposition 6.1, the PRC of Algorithm 3 is undetectable.*

Proof. The proof closely follows that of Lemma 5.13. Consider any algorithm Adv running in time $T = T(\lambda)$ and which satisfies

$$\left|\Pr_{\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda)}\left(\mathsf{Adv}^{\mathsf{Encode}(\mathsf{sk})}(1^\lambda) = 1\right) - \Pr_{\mathcal{U}}\left(\mathsf{Adv}^{\mathcal{U}}(1^\lambda) = 1\right)\right| = \nu(\lambda),$$

for some functions $\nu, T : \mathbb{N} \to \mathbb{R}_{\ge 0}$. Here $\mathcal{U}$ denotes the oracle that returns a uniformly random string of the same length as $\mathsf{Encode}(\mathsf{sk})$. We will use Adv to construct an adversary Adv' which distinguishes between the distributions $\mathcal{D}_{n,\mathbb{F}_q,C,\delta/3,T}$ and $\mathrm{Unif}((\mathbb{F}_q^n)^T)$. (We omit the dependence of the various parameters on λ.)

Note that Adv requires oracle access to an oracle $\mathcal{O}$ (to be interpreted as either $\mathsf{Encode}(\mathsf{sk})$ or the uniform distribution) which when called outputs a sample in $\mathbb{F}_q^n$. The algorithm Adv' simulates Adv as follows: first, Adv' draws $o \sim \mathrm{Unif}(\mathbb{F}_q^n)$. Then, for $t \in [T]$, for the tth call that Adv makes to the oracle $\mathcal{O}$, Adv' uses the tth sample from its distribution (either $\mathcal{D}_{n,\mathbb{F}_q,C,\eta,T}$ or $\mathrm{Unif}((\mathbb{F}_q^n)^T)$) which is an element $c \in \mathbb{F}_q^n$, to simulate the oracle call $\mathcal{O}$ by simply returning $c + o$. It is clear that the running time of Adv' is at most $T(\lambda) \cdot \mathrm{poly}(\lambda)$.

Note that, in the event that Adv' is given a sample from $\mathcal{D}_{n,\mathbb{F}_q,C,\eta,T}$, then it exactly simulates $\mathsf{Adv}^{\mathsf{Encode}(\mathsf{sk})}(1^\lambda)$, for $\mathsf{sk} = (\sigma, \pi_1, \dots, \pi_n, o)$, where $\sigma, \pi_1, \dots, \pi_n$ are the (secret) permutations used in the definition of $\mathcal{D}_{n,\mathbb{F}_q,C,\eta,T}$. This holds because the definition of $\mathsf{Encode}(\mathsf{sk})$ in Algorithm 3 exactly parallels the definition of $\mathcal{D}_{n,\mathbb{F}_q,C,\eta,T}$ in Section 3.1 (with the addition of the one-time pad o).

On the other hand, in the event that Adv' is given a sample from $\mathrm{Unif}((\mathbb{F}_q^n)^T)$, then it exactly simulates $\mathsf{Adv}^{\mathcal{U}}(1^\lambda)$ because each of the simulated oracle calls to $\mathcal{U}$ returns a uniformly random

element of $\mathbb{F}_q^n$. Thus, under the assumption that the distributions $\mathcal{D}_{n,\mathbb{F}_q,C,\eta,T}$ and $\mathrm{Unif}((\mathbb{F}_q^n)^T)$ are computationally indistinguishable to sub-exponential time algorithms, we must have that either $T(\lambda) \geq \exp\left(\lambda^{\Omega(1)}\right)$ or $\nu(\lambda) \leq \exp\left(-\lambda^{\Omega(1)}\right)$. □

Lemma 6.4. *In the setting of Proposition 6.1, the PRC of Algorithm 3 has strong adaptive robustness to any channel $\mathcal{E}$ introducing a fraction of at most $\delta/3$ of substitutions.*

Proof. Fix any sk, let us consider $x, c, \hat{c}, c'$ as computed in the execution of $\mathsf{Encode}(\mathsf{sk})$. Since $\eta = \delta/3$ is a constant, with probability $1 - \exp(-\Omega(n))$ over the randomness of the channel SC_η in $\mathsf{Encode}(\mathsf{sk})$, we have that $D_{\mathsf{H}}(c', \hat{c}) \leq \frac{\delta}{2} \cdot n$. Letting $y = c' + o$ denote the output of $\mathsf{Encode}(\mathsf{sk})$ and $y' = \mathcal{E}(y)$ for any channel $\mathcal{E}$ introducing at most a fraction of $\delta/3$ of substitutions (which can depend on sk), it follows that $D_{\mathsf{H}}(y' - o, \hat{c}) \leq \delta \cdot n$ with probability $1 - \exp(-\Omega(n))$. This in particular implies that $D_{\mathsf{H}}(y'', c) \leq \delta \cdot n$, where y'' is as computed in $\mathsf{Decode}(\mathsf{sk}, y)$ and c is as computed in $\mathsf{Encode}(\mathsf{sk})$. Thus, since Decode_C corrects up to a fraction δ of substitutions, we have that $\mathsf{Decode}(\mathsf{sk}, y') = \mathsf{True}$ with probability $1 - \exp(-\Omega(n))$, as desired. □

Algorithm 3 Substitution-robust PRC from permuted codes

Require: Code family $C = \{C(\lambda)\}_{\lambda \in \mathbb{N}}$ with $C(\lambda) \subseteq \mathbb{F}_{q(\lambda)}^{n(\lambda)}$, encoding algorithm $\mathsf{Encode}_C : \mathbb{F}_{q(\lambda)}^{k(\lambda)} \to \mathbb{F}_{q(\lambda)}^{n(\lambda)}$ and decoding algorithm $\mathsf{Decode}_C : \mathbb{F}_{q(\lambda)}^{n(\lambda)} \to \mathbb{F}_{q(\lambda)}^{k(\lambda)} \cup \{\perp\}$ for functions $q, n, k : \mathbb{N} \to \mathbb{N}$. Constant $\delta > 0$ determining noise rate. *(Dependence on λ omitted below.)*

function $\mathsf{KeyGen}(1^\lambda)$
  Let $\sigma \leftarrow \mathrm{Unif}(S_{[n]}), \pi_1, \ldots, \pi_n \leftarrow \mathrm{Unif}(S_{\mathbb{F}_q})$ be chosen uniformly at random.
  $o \leftarrow \mathrm{Unif}(\mathbb{F}_q^n)$.
  return $\mathsf{sk} := (\sigma, \pi_1, \ldots, \pi_n, o)$.
function $\mathsf{Encode}(\mathsf{sk})$
  Write $\mathsf{sk} = (\sigma, \pi_1, \ldots, \pi_n, o)$.
  Draw $x \leftarrow \mathrm{Unif}(\mathbb{F}_q^k)$, and $c \leftarrow \mathsf{Encode}_C(x) \in \mathbb{F}_q^n$.
  Define $\hat{c} \in \mathbb{F}_q^n$ by $\hat{c}_i = \pi_i(c_{\sigma(i)})$ for each $i \in [n]$.
  Let $c' \leftarrow \mathrm{SC}_{\delta/2}(\hat{c})$.
  return $y := c' + o$.
function $\mathsf{Decode}(\mathsf{sk}, y)$
  Write $\mathsf{sk} = (\sigma, \pi_1, \ldots, \pi_n, o)$.
  Let $y' := y - o$.
  Define $y'' \in \mathbb{F}_q^n$ by $y''_i = \pi_i^{-1}(y'_{\sigma^{-1}(i)})$ for each $i \in [n]$.
  Let $\hat{y} \leftarrow \mathsf{Decode}_C(y'')$.
  if $\hat{y} \in C$ and $D_{\mathsf{H}}(\hat{y}, y') \leq \delta \cdot n$ **then**
    return True
  else
    return False

Acknowledgments. We thank Yuval Ishai for suggesting that we look into the permuted puzzles assumption. We thank Vinod Vaikuntanathan for helpful discussions during the early stages of this work.

Miranda Christ was partially supported by a Google CyberNYC grant, an Amazon Research Award, and NSF grants CCF-2312242, CCF-2107187, and CCF-2212233. This work was done in part while some of the authors were visiting the Simons Institute for the Theory of Computing. Ankur Moitra was supported in part by a Microsoft Trustworthy AI Grant, NSF-CCF 2430381, an ONR grant, and a David and Lucile Packard Fellowship. Daniel Wichs was supported by NSF CNS-2349972 and CNS-2055510.

References

[AAC+25] Omar Alrabiah, Prabhanjan Ananth, Miranda Christ, Yevgeniy Dodis, and Sam Gunn. Ideal pseudorandom codes. In Michal Koucký and Nikhil Bansal, editors, *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, Czechia, June 23-27, 2025*, pages 1638–1647. ACM, 2025.

[Aar22] Scott Aaronson. My AI Safety Lecture for UT Effective Altruism. `https://scottaaronson.blog/?p=6823`, November 2022.

[AG11] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In *International Colloquium on Automata, Languages, and Programming*, pages 403–415. Springer, 2011.

[BCG+22] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Nicolas Resch, and Peter Scholl. Correlated pseudorandomness from expand-accumulate codes. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 603–633. Springer, 2022.

[BCH+25] Fabrice Benhamouda, Caicai Chen, Shai Halevi, Yuval Ishai, Hugo Krawczyk, Tamer Mour, Tal Rabin, and Alon Rosen. Encrypted matrix-vector products from secret dual codes. In Chun-Ying Huang, Jyh-Cheng Chen, Shiuh-Pyng Shieh, David Lie, and Véronique Cortier, editors, *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, CCS 2025, Taipei, Taiwan, October 13-17, 2025*, pages 394–408. ACM, 2025.

[BHJK25] Rares-Darius Buhai, Jun-Ting Hsieh, Aayush Jain, and Pravesh K. Kothari. The quasi-polynomial low-degree conjecture is false, 2025.

[BHMW21] Elette Boyle, Justin Holmgren, Fermi Ma, and Mor Weiss. On the security of doubly efficient PIR. *Cryptology ePrint Archive*, 2021.

[BHW19] Elette Boyle, Justin Holmgren, and Mor Weiss. Permuted puzzles and cryptographic hardness. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part II*, Lecture Notes in Computer Science, pages 465–493. Springer, 2019.

[BIPW17] Elette Boyle, Yuval Ishai, Rafael Pass, and Mary Wootters. Can we access a database both locally and privately? In *Theory of Cryptography: 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II 15*, pages 662–693. Springer, 2017.

[BW21] Keller Blackwell and Mary Wootters. A note on the permuted puzzles toy conjecture. *arXiv preprint arXiv:2108.07885*, 2021.

[CG24] Miranda Christ and Sam Gunn. Pseudorandom error-correcting codes. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part VI*, volume 14925 of *Lecture Notes in Computer Science*, pages 325–347. Springer, 2024.

[CGZ24] Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models. In Shipra Agrawal and Aaron Roth, editors, *The Thirty Seventh Annual Conference on Learning Theory, June 30 - July 3, 2023, Edmonton, Canada*, volume 247 of *Proceedings of Machine Learning Research*, pages 1125–1139. PMLR, 2024.

[CHR17] Ran Canetti, Justin Holmgren, and Silas Richelson. Towards doubly efficient private information retrieval. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II*, Lecture Notes in Computer Science, pages 694–726. Springer, 2017.

[CHS25] Aloni Cohen, Alexander Hoover, and Gabe Schoenbach. Watermarking language models for many adaptive users. In Marina Blanton, William Enck, and Cristina Nita-Rotaru, editors, *IEEE Symposium on Security and Privacy, SP 2025, San Francisco, CA, USA, May 12-15, 2025*, pages 2583–2601. IEEE, 2025.

[CRR21] Geoffroy Couteau, Peter Rindal, and Srinivasan Raghuraman. Silver: Silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III*, volume 12827 of *Lecture Notes in Computer Science*, pages 502–534. Springer, 2021.

[DJK25] Youlong Ding, Aayush Jain, and Ilan Komargodski. A new approach for lpn-based pseudorandom functions: Low-depth and key-homomorphic. In Michal Koucký and Nikhil Bansal, editors, *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, Czechia, June 23-27, 2025*, pages 1898–1909. ACM, 2025.

[DMR25] Nico Döttling, Anne Müller, and Mahesh Sreekumar Rajasree. Separating pseudorandom codes from local oracles. *IACR Cryptol. ePrint Arch.*, page 1020, 2025.

[FGJ+24] Jaiden Fairoze, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, and Mingyuan Wang. Publicly-detectable watermarking for language models. *IACR Commun. Cryptol.*, 1(4):31, 2024.

[FK18] Michael A. Forbes and Zander Kelley. Pseudorandom generators for read-once branching programs, in any order. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 946–955. IEEE Computer Society, 2018.

[GG25] Surendra Ghentiyala and Venkatesan Guruswami. New constructions of pseudorandom codes. In Alina Ene and Eshan Chattopadhyay, editors, *Approximation,*

Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2025, August 11-13, 2025, Berkeley, CA, USA, volume 353 of *LIPIcs*, pages 54:1–54:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025.

[GGW25] Sanjam Garg, Sam Gunn, and Mingyuan Wang. Black-box crypto is useless for pseudorandom codes. *CoRR*, abs/2506.01854, 2025.

[GM24] Noah Golowich and Ankur Moitra. Edit distance robust watermarks via indexing pseudorandom codes. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[GR07] Venkatesan Guruswami and Atri Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy, 2007.

[GRS25] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. *Essential Coding Theory*. Department of Computer Science and Engineering, University at Buffalo, SUNY, draft (2025) edition, 2025. Available online at `https://www.cse.buffalo.edu/faculty/atri/courses/coding-theory/book/`.

[GS98] Venkatesan Guruswami and Madhu Sudan. Improved decoding of reed-solomon and algebraic-geometric codes. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, FOCS '98, page 28, USA, 1998. IEEE Computer Society.

[GZS25] Sam Gunn, Xuandong Zhao, and Dawn Song. An undetectable watermark for generative image models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[Hes02] F. Hess. Computing riemann—roch spaces in algebraic function fields and related topics. *J. Symb. Comput.*, 33(4):425–445, April 2002.

[HH23] Pooya Hatami and William Hoza. Theory of unconditional pseudorandom generators. *Electron. Colloquium Comput. Complex.*, TR23-019, 2023.

[HLLL25] Xuming Hu, Hanqian Li, Jungang Li, and Aiwei Liu. Videomark: A distortion-free robust watermarking framework for video diffusion models. *CoRR*, abs/2504.16359, 2025.

[Hop18] Samuel B. Hopkins. *Statistical Inference and the Sum of Squares Method*. PhD thesis, Cornell University, Ithaca, NY, August 2018. Ph.D. dissertation.

[HvL98] Tom Høholdt and van Lint. Algebraic geometry codes. In *Handbook of Coding Theory*, pages 871–962, United Kingdom, 1998. Elsevier.

[KGW+23] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023,*

Honolulu, Hawaii, USA, volume 202 of *Proceedings of Machine Learning Research*, pages 17061–17084. PMLR, 2023.

[KTHL24] Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research*, 2024.

[KWB19] Dmitriy Kunisky, Alexander S. Wein, and Afonso S. Bandeira. Notes on computational hardness of hypothesis testing: Predictions using the low-degree likelihood ratio. *arXiv preprint arXiv:1907.11636*, 2019.

[LSS26] Huijia Lin, Kameron Shahabi, and Min Jae Song. Unforgeable watermarks for language models via robust signatures. *arXiv preprint arXiv:2602.15323*, 2026.

[O'D03] Ryan William O'Donnell. *Computational applications of noise sensitivity*. PhD thesis, Massachusetts Institute of Technology, 2003.

[SG95] Henning Stichtenoth and Arnaldo Garcia. A tower of artin-schreier extensions of function fields attaining the drinfeld-vladut bound. *Inventiones mathematicae*, 121(1):211–222, 1995.

[SKS19] Jad Silbak, Swastik Kopparty, and Ronen Shaltiel. Quasilinear time list-decodable codes for space bounded channels. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 302–333, 2019.

[SW98] M. Amin Shokrollahi and Hal Wasserman. Decoding algebraic-geometric codes beyond the error-correction bound. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, page 241–248, New York, NY, USA, 1998. Association for Computing Machinery.

[YZC+25] Zijin Yang, Xin Zhang, Kejiang Chen, Kai Zeng, Qiyi Yao, Han Fang, Weiming Zhang, and Nenghai Yu. Gaussian shading++: Rethinking the realistic deployment challenge of performance-lossless image watermark for diffusion models. *CoRR*, abs/2504.15026, 2025.

[ZALW24] Xuandong Zhao, Prabhanjan Vijendra Ananth, Lei Li, and Yu-Xiang Wang. Provable robust watermarking for ai-generated text. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[ZEF+24] Hanlin Zhang, Benjamin L. Edelman, Danilo Francati, Daniele Venturi, Giuseppe Ateniese, and Boaz Barak. Watermarks in the sand: Impossibility of strong watermarking for language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[ZGC+25] Xuandong Zhao, Sam Gunn, Miranda Christ, Jaiden Fairoze, Andrés Fábrega, Nicholas Carlini, Sanjam Garg, Sanghyun Hong, Milad Nasr, Florian Tramèr, Somesh Jha, Lei Li, Yu-Xiang Wang, and Dawn Song. Sok: Watermarking for ai-generated content. In Marina Blanton, William Enck, and Cristina Nita-Rotaru, editors, *IEEE Symposium*

on Security and Privacy, SP 2025, San Francisco, CA, USA, May 12-15, 2025, pages 2621–2639. IEEE, 2025.