Achieving Shannon Capacity for Computationally Bounded Errors

George Lu
UT Austin
gclu@cs.utexas.edu

Jad Silbak
Massachusetts Institute of Technology
jadsilbak@gmail.com

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

Abstract

We study error correction in a computationally bounded world, where errors are introduced by an arbitrary polynomial-time adversarial channel. Recent works construct *seeded* codes in this model, where the encoding and decoding procedure share a public random seed. They achieve significantly better tradeoffs between rate and error tolerance than what is possible information theoretically for unique decoding, essentially matching the parameters of the best known efficiently list-decodable codes. Over the binary alphabet, however, this is still well short of the optimal *Shannon capacity* with rate $R \approx 1 - H_2(p)$ for a $p < 1/4$ fraction of errors. Even heuristic constructions meeting this target were not previously known. We make progress towards this goal.

- **Secret-Key Codes.** We first study *secret-key* codes, where the encoder and decoder share a secret key hidden from the adversarial channel. Lipton (STACS '94) constructed *one-time secure* secret-key codes achieving Shannon capacity in this setting, but it was unknown whether one can get CPA (resp. CCA) security where the adversary may query an encoding oracle (resp. also a decoding oracle). We construct CCA-secure secret-key codes achieving Shannon capacity via *pseudorandom codes (PRCs)*.
- **Seeded Codes (Heuristic).** We can heuristically upgrade the resulting secret-key codes to seeded codes by publishing an obfuscation of the encoding/decoding procedures with a hard-coded secret key as a seed. Security holds in the ideal obfuscation model.
- **Public-key Codes.** We also consider *public-key* codes, where the decoder has a secret key and the encoder has the corresponding public key. We construct such CPA-secure public-key codes achieving Shannon capacity with unique decoding for $p < 1/4$ errors, and list decoding all the way to $p < 1/2$ errors, assuming PRCs and the subexponential security of standard crypto assumptions (e.g., LWE or DDH or QR or DCR). We also show how to get CCA security in the random oracle model.

Our secret-key and public-key codes meeting Shannon capacity are also simultaneously pseudorandom codes.

*Research supported by NSF grants CNS-2349972 and CNS-2055510.

Contents

1 Introduction **3**
- 1.1 Our Results . . . 4

2 Technical Overview **6**

3 Preliminaries **9**
- 3.1 Standard Error-Correcting Codes . . . 9
  - 3.1.1 List-Decodable and List-Recoverable Codes . . . 10
  - 3.1.2 Codes for Permuted Errors . . . 10
- 3.2 Secret-Key and Public-Key Codes, With CPA and CCA Security . . . 11
  - 3.2.1 Known Results for Pseudorandom Codes . . . 13
- 3.3 Correlation-Intractable Hashing . . . 14

4 Shannon-Rate Secret Key Codes **15**
- 4.1 Proof of Lemma 4.3 . . . 16

5 Heuristic Seeded Codes via Ideal Obfuscation **19**

6 List Decodable Pseudorandom Codes With Optimal Robustness **20**
- 6.1 Polynomial Message Space List-Decodable Pseudorandom Codes . . . 20
- 6.2 From Polynomial Message Space to Exponential Message Space . . . 21
- 6.3 Proof of Theorem 6.1 . . . 23

7 Public Key Shannon Rate Codes **24**
- 7.1 Construction for Optimal Rate CPA List-Decodable Codes . . . 24
  - 7.1.1 Proof of Lemma 7.4 . . . 25
- 7.2 Optimal Rate CPA Uniquely Decodable Codes in the Plain Model . . . 29
  - 7.2.1 Decomposable Binary Linear Codes with Pairwise Column-Robustness . . . 29
  - 7.2.2 Correlation Intractability Against Permuted and Masked Linear Codes . . . 30
  - 7.2.3 Construction and Proof of Theorem 7.9 . . . 32
- 7.3 Optimal Rate CCA Codes . . . 34
  - 7.3.1 Achieving Simulation-Decodability . . . 35
  - 7.3.2 Proof of Lemma 7.26 . . . 36
  - 7.3.3 From Public-Key CPA Codes With Simulation Decoding to CCA Codes in the RO Model . . . 39
  - 7.3.4 Proof of Lemma 7.30 . . . 39

8 Acknowledgements **47**

A Shannon-Rate Linear Codes with Pairwise Column-Robustness **50**

1 Introduction

Coding Theory. The theory of error correction originates in the seminal works of Shannon [Sha48] and Hamming [Ham50]. In its most basic form, a sender encodes a message $m \in \{0,1\}^k$ into a codeword $c = \mathsf{Enc}(m) \in \{0,1\}^n$ and transmits c over a noisy channel that may corrupt some of the n transmitted bits. A receiver observes a possibly corrupted value $c' \in \{0,1\}^n$ and runs a decoding procedure $\mathsf{Dec}(c')$ that should uniquely recover the original message m.[1] Shannon's work considered *random errors* (i.e., the binary symmetric channel (BSC)), where the channel flips each codeword bit with some probability p, while Hamming considered *worst-case / adversarial* errors, where the channel can flip an arbitrary subset of pn codeword bits. The main parameters of interest are the *rate* of the code, which is defined as $R = k/n$ and measures how much data can be packed into the codeword, and the *relative error tolerance* p of the code, which measures the fraction of errors that the code can correct. There is an inherent tension between these parameters; higher error tolerance p typically requires more redundancy and hence smaller rate R. The goal is to achieve optimal tradeoffs between them.

Information-Theoretic Bounds. In Shannon's model, the optimal tradeoffs between rate and error-tolerance are well understood. It is possible to correct random errors for any error probability $p < 1/2$ with rate $R \approx 1 - H_2(p)$, where $H_2(p) := p \log_2 \frac{1}{p} + (1-p) \log_2 \frac{1}{1-p}$ is the binary entropy function.[2] We refer to the above rate as the *Shannon capacity*. Moreover, we have efficient constructions that achieve Shannon capacity and are guaranteed to recover the original message with overwhelming probability.

In Hamming's model, the optimal tradeoffs are still not fully understood, despite decades of study. It is known that unique error correction is only possible for $p < 1/4$ fraction of errors. The best achievable rate *existentially* (i.e., derived via the probabilistic method without an efficient construction) is given by the Gilbert-Varshamov (GV) bound $R \approx 1 - H_2(2p)$. In other words, this only allows us to correct half as many errors compared to Shannon's model for any given rate. It is widely conjectured that this is asymptotically optimal, but we do not actually know how to prove this. Moreover, we also do not know how to achieve the GV bound efficiently, and the best known efficient constructions for the Hamming model only achieve the somewhat worse *Zyablov bound* [Zya71].

In the Hamming model, we can also consider a relaxation of unique decoding called *list-decoding*, where the decoding algorithm only needs to output a polynomial-size list of messages that includes the original message. Existentially, it is known that list-decoding is achievable for $p < 1/2$ fraction of errors with a rate matching Shannon capacity $R \approx 1 - H_2(p)$, which is optimal. However, the best known efficiently list-decodable codes only achieve the somewhat worse *Blokh-Zyablov (BZ) bound* [BZ76].

While the formulations of the Zyablov and BZ bounds are complicated, we give numeric examples in Table 1.

Computationally Bounded Errors. Following the usual paradigm in cryptography, it is natural to assume that the channel is adversarial but *computationally bounded*, meaning that the error pattern it introduces must be computable in polynomial time. This model lies between Shannon's model of random errors and Hamming's model of (computationally unbounded) worst-case errors. While, for all intents and purposes, it is as good as Hamming's model in a computationally bounded world, it may allow for parameters that match those of Shannon's much weaker model. Indeed, a long line of works shows that one can achieve improved parameters in this model over Hamming's. For the corresponding codes, there must exist error patterns that the code fails to correct, but no efficient adversary is able to find them. In order to achieve this, we also need to relax the coding syntax, where the encoding/decoding algorithms are public deterministic procedures – otherwise a non-uniform efficient adversary can always just hard-code a worst-case error pattern for which error correction fails. Several such relaxations were considered in prior works, and are surveyed below.

Secret-key codes. Lipton [Lip94] introduced a model where the encoder $\mathsf{Enc}_{\mathsf{sk}}$ and decoder $\mathsf{Dec}_{\mathsf{sk}}$ share a secret key sk hidden from the channel. In this setting, it is possible to achieve Shannon capacity $R \approx 1 - H_2(p)$ for any $p < 1/2$ via a simple construction. In fact, this can be even accomplished information-theoretically against computationally unbounded adversaries with a sufficiently large shared key, and computational security is only needed to shrink the key size. A major limitation of this result is that it only provides *one-time security*, analogously to one-time

[1] One can also consider coding over larger alphabets, but in this work we focus on the binary alphabet.
[2] Throughout the introduction, we use $\approx$ to hide additive constants that we can set arbitrarily close to 0.

pad encryption, and security does not hold if an adversary can observe multiple codewords generated using the same key. Ideally, we would also like to achieve *chosen-plaintext attack (CPA)* (resp. *chosen-codeword attack (CCA)*) security, where the adversary additionally gets access to the encoding oracle $\mathsf{Enc}_{\mathsf{sk}}$ (resp. also the decoding oracle $\mathsf{Dec}_{\mathsf{sk}}$), analogously to CPA/CCA secure symmetric-key encryption. For CPA security, it is easy to see that one cannot uniquely correct $p > 1/4$ fraction of errors with rate $R > 0$.[3] But even for $p < 1/4$, Lipton's result does not readily extend to provide CPA security.

Public-key Decoding. Micali, Peikert, Sudan, and Wilson [MPSW10] proposed a public-key analogue, where the encoder $\mathsf{Enc}_{\mathsf{sk}}$ holds a secret key sk while the decoder (and the channel) get a corresponding public key pk. Assuming just digital signatures (one-way functions), for any error tolerance $p < 1/2$ they construct a uniquely decodable code with a rate R that matches that of *best known efficiently list-decodable code* with error tolerance p. In particular, the rate of these codes meets the BZ bound. While they consider a setting where the adversary has access to the encoding oracle, their result only holds if the encoder is stateful and the adversary only gets access to the encoding oracle before seeing the challenge codeword but not after. If we remove these restrictions, then the maximal achievable tolerance becomes $p < 1/4$.[4]

Seeded codes. Grossman, Holmgren, and Yogev [GHY20] proposed *seeded codes*, where the encoder Enc_s and decoder Dec_s only share a random *public seed* s that is also known to the adversarial channel. They construct such codes achieving the BZ bound for any $p < 1/4$ in the random oracle model. Subsequent works of Lu, Silbak and Wichs [SW25b, SW25a, LSW25] study seeded codes in the standard model, and the state of the art [LSW25] achieves the BZ bound for $p < 1/4$ under standard cryptographic assumptions such as DDH or LWE or QR or DCR.

Additional Models. The work of [SW25b] also considers *randomized codes* without any shared seed, where the encoding algorithm uses fresh local randomness on each invocation. However, they only construct these for large alphabets and it remains open to get constructions over a binary alphabet. The works of [GS16, SS21a, SS22, SS24b] consider a setting where the complexity of the code can exceed that of the adversarial channel. For example the work of [SS24b] shows that for $p < 1/4$ and any polynomial n^c, there is a randomized code with polynomial complexity $n^d > n^c$ that achieves Shannon capacity $R \approx 1 - H_2(p)$ for channels running in time n^c. In contrast, in this work we focus on the standard cryptographic setting where the codes must run in some fixed polynomial time, and can protect against arbitrary polynomial time channels, including ones whose run-time vastly exceeds that of the code.

Summary of the Remaining Gaps. To summarize, existing results demonstrate that codes for computationally bounded errors can circumvent classical information-theoretic barriers. Under standard cryptographic assumptions, prior work constructed seeded codes in this setting with unique decoding, where the rate matches that of the best known efficiently list-decodable codes, for error tolerance $p < 1/4$. Unfortunately, the rate of the currently best known efficiently list-decodable codes only achieves the BZ bound, which is significantly worse than the optimal goal of reaching Shannon capacity. While one may hope that future improvements in list-decodable codes will allow us to efficiently reach Shannon capacity, this would require breaking through a long-standing barrier in coding theory. Without such a breakthrough, prior binary codes for computationally bounded channels are stuck at the BZ bound, even if we consider relaxations of seeded codes, such as public-key codes or symmetric-key codes with CPA security. Reaching Shannon capacity for any such codes was previously unknown even heuristically.

1.1 Our Results

In this work, we make progress towards constructing codes for computationally bounded errors and achieving Shannon capacity.

Secret-Key Codes. We first consider *secret-key* codes $(\mathsf{Enc}_{\mathsf{sk}}, \mathsf{Dec}_{\mathsf{sk}})$, where the encoder and decoder share a secret key hidden from the adversarial channel. Previously, Lipton [Lip94] gave a simple construction with *one-time security* achieving Shannon capacity, but it was unknown whether one can get CPA (resp. CCA) security

[3] If an adversary observes two random codewords, their relative Hamming distance is likely to be $< 1/2$. The adversary can therefore choose a mid-point that is of relative distance $< 1/4$ from each one, and decoding will fail to uniquely output at least one of the two starting messages.

[4] It appears likely, although not entirely trivial, that the result of [MPSW10] can be extended to achieve the BZ bound for $p < 1/4$ without the above restrictions.

where the adversary may also query an encoding oracle (resp. also a decoding oracle). We construct CCA-secure secret-key codes that meet Shannon capacity for $p < 1/4$. We do so via the use of *pseudorandom codes (PRCs)* [CG24, GM24, GG25, AAC+25, CGG+26], which were originally introduced in a seemingly completely unrelated context of watermarking large language models. A PRC is a secret-key encoding scheme that is simultaneously: (i) pseudorandom, in that any polynomial number of codeword are computationally indistinguishable from uniform, and (ii) error-correcting or robust in that decoding succeeds even when errors are introduced. We rely on pseudorandom codes that can tolerate any $p < 1/4$ rate of errors (but can have arbitrarily poor rate). Such PRCs are known assuming $2^{O(\sqrt{n})}$-hardness of the *learning parity with noise (LPN)* assumption [CG24], or the *permuted codes assumption* [CGG+26].

Seeded Codes. We observe that CCA-secure secret-key codes can be converted into seeded codes by obfuscating the encoding/decoding procedures (with a hard-coded secret key) and publishing the obfuscated programs as the seed. Security trivially holds in the *ideal obfuscation model*, where we replace the obfuscated programs by black-box oracles. The work of [JLLW23] showed that one can instantiate ideal obfuscation in a variant of the random oracle model called the *pseudorandom oracle (PRO) model*, assuming indistinguishability obfuscation, and therefore we also get such codes in the PRO model. Overall, this provides strong heuristic evidence that we can securely instantiate our construction. In particular, it gives the first heuristic candidate for seeded codes achieving Shannon capacity for any $p < 1/4$.

Public-key Codes. We next consider *public-key* codes $(\mathsf{Enc}_{\mathsf{pk}}, \mathsf{Dec}_{\mathsf{sk}})$, where the decoder has a secret key sk and the encoder has the corresponding public key pk.[5] This notion lies in between symmetric-key codes and seeded codes. We first show how to construct such *list-decodable* CPA-secure public-key codes that achieve Shannon capacity for any $p < 1/2$ fraction of errors. We do so via public-key PRCs, which can in turn be constructed assuming $2^{O(\sqrt{n})}$-hardness of the *learning parity with noise (LPN)* assumption [CG24]. We then convert the above list-decodable codes into *uniquely-decodable* CPA-secure public-key codes that achieve Shannon capacity for any $p < 1/4$ fraction of errors via the use of correlation-intractable (CI) hashing, relying on the paradigm of [SW25a, LSW25]. The needed CI hash functions can be instantiated from the subexponential security of standard crypto assumptions such as DDH or LWE or QR or DCR. Finally we show to make the corresponding list-decodable or uniquely-decodable codes CCA secure in the random oracle model.

Pseudorandomness. While this was not our main goal, we also show that our constructions of CPA/CCA secret-key and public-key codes achieving Shannon capacity are also simultaneously *pseudorandom codes* themselves. In particular, the codewords in these constructions look pseudorandom even to an adversary with the corresponding encoding/decoding oracles. Therefore we get corresponding pseudorandom codes that achieve Shannon capacity.

Discussion of Parameters. In Table 1 we give some numeric examples of the best known rates achievable in different error models. Note that, for information-theoretic unique decoding for worst-case errors, the rate R goes to 0 as error tolerance p approaches .25 from below, and this is inherent. Interestingly, for computationally bounded errors, both prior works and our results in rows 4 and 5 achieve some positive rate R as p approaches .25 from below (in our case $R \approx .188$), but then the rate drops to 0 at $p > .25$, meaning that the graph is not continuous at $p = .25$. (This was already observed in prior works.) To see the dramatic differences in parameters between the different models, consider a high-error regime where we want to correct a 20% fraction of errors. The best prior efficient information-theoretic codes had rate $R < .001$, meaning that the codeword had to be $> 1,000\times$ larger than the message! The best prior codes for computationally bounded errors had rate $R = .073$ meaning that the codeword had to be $> 13\times$ larger than the message. Our results achieve $R = .278$ meaning that the codeword only has to be $< 3.6\times$ larger than the message.

[5] This is the reverse of the *public-key decoder* codes of [MPSW10] where the encoder had a secret key and the decoder had the corresponding public key.

Model	$p = 0.05$	$p = 0.10$	$p = 0.15$	$p = 0.20$
(1) Random Errors	0.713	0.531	0.390	0.278
(2) Worst-Case Errors: Existential Construction	0.531	0.278	0.118	0.029
(3) Worst-Case Errors: Efficient Construction	0.141	0.041	0.009	< 0.001
(4) Comp. Bounded Errors: Prior Work	0.415	0.233	0.132	0.073
(5) Comp. Bounded Errors: Our Results	0.713	0.531	0.390	0.278

Table 1: Best known rates R for unique decoding for different error models (rows) and error-tolerance p (columns). *(1) is the Shannon bound $R = 1 - H_2(p)$, (2) is the GV bound $R = 1 - H_2(2p)$, (3) is the Zyablov bound for distance $2p$, (4) is the Blokh-Zyablov bound, and (5) is the Shannon bound. For rows 2-5, the rate drops to 0 for $p > .25$. Prior works for Row (4) include [MPSW10, GHY20, SW25b, SW25a, LSW25]. Rows 1 and 5 are known to be optimal for the given model.*

2 Technical Overview

One-Time Secure Secret-Key Codes. We start by recalling the one-time secure secret-key codes of Lipton [Lip94]. The construction is very simple. The encoder and decoder share a random secret key consisting of a permutation $\pi : [n] \to [n]$ and an n-bit one-time pad. The encoder first encodes the data with an optimal code for *random errors*, which can correct from a random subset of pn flips with overwhelming probability. This is a small variant of the binary symmetric channel considered by Shannon, and we have efficiently computable codes for this setting with optimal rate $R \approx 1 - H_2(p)$ for $p < 1/2$. Let $\hat{c} \in \{0,1\}^n$ be the resulting codeword. The encoder then permutes the bits of the codeword with the permutation π and applies a one-time pad, resulting in the final codeword $c \in \{0,1\}^n$. Intuitively, the one-time pad completely hides the permutation, which then translates worst-case adversarial errors on c into random errors on $\hat{c}$.

Unfortunately, the above construction is completely insecure under a CPA attack where the adversary can observe many codewords of chosen messages. Eventually it can learn the one-time pad and entire permutation π (e.g., an encoding of the all 0 message reveals the pad, and encodings of the n unit vectors then fully reveal the permutation). One may be tempted to apply the usual trick of converting a one-time secure scheme into a many-time secure scheme by using a pseudorandom function (PRF) whose key is shared by the encoder and decoder. The encoder would choose a fresh random nonce r each time and then use $PRF(r)$ to derive a fresh one-time pad for the one-time scheme. This would work if the decoder could reliably get r, but how can we securely transmit it in the face of errors, and prevent the adversary from removing it by strategically focusing its errors?

CPA-Secure Secret-Key Codes. To get around the above issues, our construction relies on a powerful recent cryptographic primitive called *pseudorandom codes (PRCs)* [CG24, GM24, GG25, AAC+25, CGG+26], originally introduced in the seemingly unrelated context of watermarking LLM outputs. A PRC is a symmetric-key or public-key encoding with two properties: (1) *robustness*: it can correctly decode from up to a p-fraction of adversarial errors, (2) *pseudorandomness*: no polynomial-time adversary can distinguish between the encoding of a chosen message and a uniformly random and independent string. We can consider CPA or CCA secure PRCs depending on whether the adversary has access to the encoding or also decoding oracles. The work of [CG24] constructed public-key PRCs under the sub-exponential LPN assumption, and the work of [GG25] did so under various "planted structure" assumptions. One downside is that these constructions can be broken in quasi-polynomial time. The recent work of [CGG+26] gave an alternate construction of symmetric-key PRCs with plausibly sub-exponential security under a "permuted codes" assumption, studied previously in the context of doubly efficient private information retrieval. We will rely on symmetric-key CPA-secure PRCs that can tolerate up to $p < 1/4$ errors, but we will only use them to encode small messages and hence will not care about the *rate* of the PRC.

Using such a PRC, we can construct a CPA-secure secret-key code that achieves Shannon capacity for up to a p

fraction of computationally bounded errors. The encoder and decoder share the secret key of a PRC along with a random permutation π. On each invocation, the encoder chooses a fresh random seed s for a pseudorandom generator (PRG) G that it expands into a long one-time pad $y = G(s)$. It encodes the seed s with a PRC resulting in a small codeword c_{PRC}. As in Lipton's construction, it encodes the message with an optimal code for *random errors* that meets Shannon capacity yielding a codeword c_{RE}. Finally it permutes the bits of the tuple $(c_{PRC}, c_{RE} \oplus y)$ via the permutation π to yield the final codeword c. The decoder gets some erroneous codeword c', undoes the permutation to recover (c'_{PRC}, z'), decodes c'_{PRC} to recover s' and the corresponding $y' = G(s')$, $c'_{RE} = z' \oplus y'$, and finally decodes c'_{RE} to recover the message.

The security analysis relies on the fact that, by the pseudorandomness of the PRC and the PRG security, CPA queries to the overall code generate responses that look uniformly random even given the permutation π. Therefore, the error pattern chosen by the adversary for the challenge codeword c is computationally independent of π. This ensures that, when decoding c, we can guarantee:

- Since the overall errors are randomly scattered by the permutation π, the relative distance between c'_{PRC} and c_{PRC} is $\approx p$ and therefore the robustness of the PRC ensures that c'_{PRC} correctly decodes to s.
- The error pattern on the codeword $c'_{RE} = z' \oplus G(s)$ is randomly permuted by π, and therefore the codeword correctly decodes to the original message by error-correction for random errors.

Furthermore the above construction preserves pseudorandomness and therefore yields CPA-secure secret-key PRCs that achieve Shannon capacity. We note that the above construction is similar to a transformation of [AAC+25] to amplify the rate of a PRC. However, the goal there is to achieve a rate that matches information-theoretic codes for worst-case errors, while here we match information-theoretic codes for random errors! In particular, we make use of the fact that the adversary cannot control the implicit error pattern on the underlying codeword c_{RE} since it is scrambled by an unknown permutation π, and therefore resilience to random errors suffices.

CCA-Secure Secret-Key Codes. The above achieves CPA security. The recent work of [AAC+25] shows how to generically upgrade CPA-secure secret-key PRCs to CCA-secure ones while preserving the rate and error tolerance. Therefore, we can generically apply their transformation to achieve CCA-secure secret-key codes (and, in fact, PRCs) that achieve Shannon capacity for any $p < 1/4$ errors.

Seeded Codes. As mentioned earlier, CCA-secure secret key codes can trivially be converted into seeded codes by obfuscating the encoding and decoding procedures (with the hard-coded secret key) using ideal obfuscation and publishing the obfuscated program as the seed. Ideal obfuscation can then be instantiated heuristically using an indistinguishability obfuscator. Overall, this gives the first heuristic construction of seeded codes that achieve Shannon capacity for any $p < 1/4$ errors.

Public-Key Codes. As an intermediate primitive between symmetric-key and seeded codes, we also consider public-key codes, where the decoder has a secret key sk and the encoder has a corresponding public key pk. We construct these from public-key PRCs while avoiding the use of ideal obfuscation that was needed for seeded codes. Let us first consider CPA security.

Our previous construction template, even if we were to instantiate it with public-key PRCs, would only yield a secret-key code since the encoder and decoder need to share a secret permutation π. Can we avoid this shared secret? We can think of π as performing two distinct jobs: (1) firstly, it randomly permutes the errors on c_{RE}, translating worst-case errors to random errors, (2) secondly, it hides the positions that contain c_{PRC}, preventing the adversary from concentrating its errors there. We can perform the first job without a shared secret, by simply deriving the permutation to apply on c_{RE} from the PRG seed s encoded in the PRC. But we cannot do this for the second job since, without knowing where c_{PRC} is first, we cannot hope to decode it. At the same time, we cannot put c_{PRC} in any fixed location since then the adversary could concentrate its errors there.

Our solution is for the encoder to create many random PRC codewords, all encoding the same PRG seed s, and sprinkle these whole PRC codewords into random chunks of our overall codeword. This approach of hiding a small but crucial piece of information inside a much larger codeword via pseudorandom-looking encodings

already appears in the work of Guruswami and Smith [GS16], and was further used in several follow-up works [SS21a, KSS19, SS21b, SS22, SS24a]. Our construction is inspired by this line of work.[6]

More concretely, we first encode the message with an optimal code for random errors that meets Shannon capacity yielding a codeword c_{RE}. We choose a PRG seed s and expand it to derive $(\pi, y) = G(s)$ consisting of a permutation π and a pad s. Let $\hat{c}_{RE} := \pi(c_{RE} \oplus y)$ denote the padded and permuted bits (abusing notation). We then create t fresh PRC encodings of s, where t can be small (e.g., security parameter). If each PRC codeword is of some small size m, we think of the n-bit codeword $\hat{c}_{RE}$ as consisting of n/m chunks of m bits each. We choose a random subset of t of these chunks at random and replace them with the corresponding PRC codewords while keeping the remaining chunks intact to derive our final codeword c. The decoder tries to decode each chunk one by one via the PRC decoding algorithm. If it succeeds to recover some (π', y'), it uses it to recover a candidate c'_{RE}, which it finally decodes via the code for random errors to recover the message. Overall, the decoding procedure may output a list of up to n/m messages, one per chunk.

As a first step, we argue that this gives us a good *list-decodable code*, meaning that the correct message is likely to be in the list. We rely on the pseudorandomness of the PRC (and the PRG) to argue that the adversary does not learn anything about the permutation π or the locations of the t chosen chunks in the challenge codeword. Therefore, for any adversarial choice of pn errors:

- With overwhelming probability at least one of the t chosen chunks gets hit with at most $\approx pm$ errors and therefore decodes correctly.
- For that chunk, the corresponding candidate c'_{RE} is hit with errors randomly permuted by π and therefore decodes to the original message with overwhelming probability by error-correction for random errors.[7]

This gives us CPA-secure list-decoding for $p < 1/4$ errors achieving Shannon capacity, which is already interesting. However, in this range we would like to get unique decoding.

To get uniqueness, we need to ensure that there is only one valid candidate in the list. We rely on the techniques of [SW25a, LSW25], which show how to convert certain list-decodable codes into uniquely decodable codes using a form of correlation-intractable (CI) hashing, which can in turn be instantiated under standard crypto assumptions. The high level idea is to encode $(x, h(x))$ where h is an appropriate CI hash function, and show that the adversary cannot find $x \neq x'$ where the encodings of $(x, h(x))$ and $(x', h(x'))$ are sufficiently close to possibly show up in the same list. Unfortunately, we cannot use that result as a black box. It operates in the setting where the underlying list-decodable code is a traditional deterministic linear code with some additional mild properties. However, our Shannon-rate list-decodable code Construction 7.3 is both randomized and non-linear. Nevertheless, we manage to show how to make this approach work, by going under the hood of the proof in [LSW25]. In particular, we show how to use the techniques of [LSW25] to achieve a stronger and more general form of correlation intractability that suffices for our application.

Lastly, we follow the work of [AAC+25] to convert CPA-secure public-key PRCs into CCA-secure ones in the Random Oracle model while preserving the rate and error-tolerance. It turns out that in the random oracle model, we can also achieve uniqueness more easily, allowing us to avoid the reliance on CI hashing.

Public-Key List-Decodable Codes for $p < 1/2$. In the above construction, we first construct CPA-secure list-decoding achieving Shannon capacity for any $p < 1/4$ errors. Ideally, if we only want list-decoding, we could go all the way to any $p < 1/2$. The restriction to $p < 1/4$ came from the underlying (low-rate) PRCs, which can only be robust to $p < 1/4$ errors. But we observe that the construction works just as well if we relax PRC robustness to only require list-decodability rather than unique decodability. So if we plug in (low-rate) PRCs that are list-decodable for any $p < 1/2$ errors then our overall construction achieves list-decoding at Shannon capacity for any $p < 1/2$ as

[6] While the high-level construction is similar to Guruswami and Smith [GS16] and subsequent follow-up works, our setting is different. In the Guruswami–Smith setting, the adversary is computationally weaker than the encoding and decoding procedures, so such hiding can be achieved without a secret key. In our setting, by contrast, the adversary is computationally stronger than the encoding and decoding procedures, which is why we rely on the cryptographic notion of pseudorandom codes. This distinction leads to different challenges in both the construction and the analysis.

[7] This includes the adversarial errors, as well as the small number of additional errors created when we replace some chunks of $\hat{c}_{RE}$ with PRC codewords.

well. We also show how to upgrade the resulting CPA-secure list-decodable codes to CCA-secure ones in the random oracle model via the same transformation as for unique decoding.

Public-Key PRCs with High Error Tolerance. The last missing component in the above public-key constructions (for both unique and list decoding) is a PRC that is list-decodable with high error-tolerance – at least up to $p < 1/4$ to get our optimal unique decoding result, and ideally all the way to $p < 1/2$ to also get our optimal list decoding result. For this part, we do not care about the *rate* of the PRCs, and the codeword can be polynomially larger than the message. We rely on the transformations above, which only use such PRCs to encode small messages, to reach Shannon capacity. Prior work [AAC+25] achieved unique decoding for any $p < 1/4$ in the secret-key setting, but in the public-key setting they only achieved this for much lower values of $p < 1/32$, and they did not consider list-decoding. We show how to get such list-decodable public-key PRCs up to the optimal $p < 1/2$ errors.

The previous works of [CG24, AAC+25] construct public-key *zero-bit PRCs* that have robustness for up to $p < 1/2$ errors. In a zero-bit PRC the encoding procedure does not take any message, and the decoding procedure only needs to detect whether the received value is close to a codeword or not. These satisfy *pseudorandomness*, where codewords look uniformly random even given the public key, and *robustness*, ensuring that an adversary given a public-key and a random codeword cannot induce a p fraction of errors that cause decoding to fail to detect. They also satisfy a *soundness* (non-triviality) property, which says that most arbitrary strings are not detected as codewords – for any string c chosen a-priori, decoding only detects with negligible probability over the random choice of the public/secret keys.

Our first step is to upgrade from a zero-bit PRC with error tolerance $p < 1/2$ to a list-decodable PRC with the same error tolerance p, but only for a polynomial-size message space (i.e., messages of logarithmic bit length). If the messages are values in $[q]$ for some polynomial q, then we choose q independent public/secret key pairs for a zero-bit PRC. To encode the message $i \in [q]$ we sample a codeword using the i'th public-key. The decoder tries decoding with all q secret keys and outputs the list of values $i \in [q]$ for which decoding detects. Robustness of the underlying zero-bit PRC ensures that the correct message is in the list. We use soundness to argue that the list size ℓ cannot be too large, and is much smaller than the full message space $[q]$. This is because the probability of any string being detected by a specific subset of ℓ codes is ε^ℓ for a negligible soundness error ε and then we can do a union bound over all such sets and all strings to argue that no string can be in more than ℓ lists.

Our next step is to upgrade from a list-decodable PRC with error tolerance p for a polynomial-size message space q into one for arbitrary exponential-size message space (i.e., arbitrary polynomial bit-length messages). We do so by relying on *list-recoverable codes* over a polynomial alphabet $[q]$. Such codes can decode by being given small input-lists of size $\ell \ll q$ for each alphabet symbol of the codeword, and are guaranteed to recover a polynomial-size output-list that contains the correct message as long as sufficiently large fraction (e.g., any constant fraction) of the input-lists contain the correct codeword symbol. We first encode the message with a list-recoverable code and then encode each codeword symbol in $[q]$ with the previously constructed PRC. If the overall error rate is some constant $p < 1/2$ then a constant fraction of small PRC codewords will have error-rate bounded by some constant $p' < 1/2$ and PRC decoding will ensure that the correct codeword symbol appears in the corresponding lists, allowing the decoding of the list-recoverable code to succeed.

3 Preliminaries

Notation We use $\Delta(x_0, x_1)$ to refer to the Hamming distance between $x_0 \in \{0,1\}^n$ and $x_1 \in \{0,1\}^n$ and $\delta(x_0, x_1) = \Delta(x_0, x_1)/n$ to be the relative hamming distance. We omit the second argument $(\Delta(x), \delta(x))$ to refer to the hamming *weight* of a single string (i.e. hamming distance to 0^n). For a permutation $\pi : [n] \to [n]$ and a string $x = (x_1, \ldots x_n) \in \{0,1\}^n$, we abuse notation and let $\pi(x)$ denote a permutation on the bits of $(x_{\pi(1)}, \ldots, x_{\pi(n)})$. For a set S, let $x \leftarrow S$ denote that x was drawn uniformly from S.

3.1 Standard Error-Correcting Codes

Definition 3.1 (Standard Error-Correcting Codes)**.** A standard error-correcting code on message space $\{0,1\}^k$ and codeword space $\{0,1\}^n$ is a set of deterministic algorithms $(\mathsf{Enc}, \mathsf{Dec})$ with the following syntax:

- $\mathsf{Enc}(x) \to c$: On input message $x \in \{0,1\}^k$, the encoding algorithm outputs codeword c.
- $\mathsf{Dec}(c') \to x$: On input string $c' \in \{0,1\}^n$, the decoding algorithm outputs a message x (or $\perp$).
- **Distance:** We define the distance of a code as the minimum relative hamming distance between two codewords

$$\min_{x \neq x' \in \{0,1\}^k} \delta(\mathsf{Enc}(x), \mathsf{Enc}(x')).$$

- **Rate:** We measure the rate R of the code as the ratio $\frac{k}{n}$.

3.1.1 List-Decodable and List-Recoverable Codes

Definition 3.2 (List-Decodable Codes [Eli57, Woz58])**.** For parameters $p \in [0, 1/2), L \in \mathbb{N}$, we say a code Π_{code} with message space $\{0,1\}^k$ and codeword space $\{0,1\}^n$ is (p, L) list-decodable if there exists additional algorithm ListDec:

- $\mathsf{ListDec}(c') \to T$ On input string $c' \in \{0,1\}^n$, the list decoding algorithm outputs a list T of messages which satisfy the following properties:
  - The size of the output list $|T| \leq L$.
  - For any message m, if there exists codeword $c \in \{0,1\}^n$ such that $\delta(c', c) \leq p$, then $m \in T$.

Observe that unique decoding is the special case of list decoding with list size $L = 1$, in which there exists at most one codeword within relative distance p of any received word.

Definition 3.3 (List-Recoverable Codes [GI01, GS98])**.** For parameters $\alpha \in [0,1], \ell, L \in \mathbb{N}$, we say a code $\Pi_{\mathsf{code}} = (\mathsf{Enc}, \mathsf{ListRec})$, such that $\mathsf{Enc} : \{0,1\}^k \to [q]^n$ is (α, ℓ, L) list-recoverable if the algorithm ListRec satisfies the following:

- $\mathsf{ListRec}(S_1, \cdots, S_n) \to T$: On input sets $S_i \subseteq [q]$ such that $|S_i| \leq \ell$ for each index $i \in [n]$, the list recovery algorithm outputs a list T of messages which satisfy the following properties:
  - The size of the output list $|T| \leq L$.
  - The list T contains all messages $m \in \{0,1\}^k$, such that $\Pr_{i \leftarrow [n]}[\mathsf{Enc}(m)_i \in S_i] \geq \alpha$.

We note that list-recoverable codes are a strict generalization of list-decodable codes (which correspond to the special case of $\ell = 1$).

3.1.2 Codes for Permuted Errors

In our constructions we require codes that recover from binary symmetric noise (BSC), that approach the Shannon capacity $R(p) = 1 - H(p)$. In fact, we require a slightly stronger property in which the decoding recovers from permuted errors (not just random and independent errors). We will use the following theorem taken from [SS22] that was used in previous results. On a high level, the code below is constructed using standard code concatenation where the outer code is a linear code with distance ε and rate $1 - \varepsilon$, and the inner code is a BSC code found in brute force search. The main observation is that random independent noise and randomly permuted fixed noise string are locally statistically close, which implies that the small inner codewords recover from permuted errors.

Theorem 3.4 (Codes for permuted errors, [Smi07, GS16, SS21a, KSS19, SS22])**.** *Let $R(p) = 1 - H(p)$. For every constant $0 \leq p < \frac{1}{2}$, and every sufficiently small constant $\varepsilon > 0$, there exists functions $\mathsf{Enc}_{\mathsf{BSC}} : \{0,1\}^{k=(R(p)-\varepsilon)\cdot n} \to \{0,1\}^n$, $\mathsf{Dec}_{\mathsf{BSC}} : \{0,1\}^n \to \{0,1\}^k$ such that:*

- *For every $m \in \{0,1\}^k$, and error pattern $e \in \{0,1\}^n$, such that $\delta(e) < p$, let $\pi : [n] \to [n]$ be a random and independent permutation,*

$$\Pr[\mathsf{Dec}_{\mathsf{BSC}}(\mathsf{Enc}_{\mathsf{BSC}}(m) + \pi(e)) = m] > 1 - 2^{-\Omega(n^{0.1})}$$

Where the probability is taken over the random choice of π from the set of all possible permutations.

- $(\mathsf{Enc}_{\mathsf{BSC}}, \mathsf{Dec}_{\mathsf{BSC}})$ *are explicit.*
- $(\mathsf{Enc}_{\mathsf{BSC}}, \mathsf{Dec}_{\mathsf{BSC}})$ *has "monotone decoding" meaning that for every* $m \in \{0,1\}^k$ *and every* $e, e' \in \{0,1\}^n$ *such that for every* $i \in [n]$, $e_i \leq e'_i$,

$$\mathsf{Dec}_{\mathsf{BSC}}(\mathsf{Enc}_{\mathsf{BSC}}(m) \oplus e') = m \Rightarrow \mathsf{Dec}_{\mathsf{BSC}}(\mathsf{Enc}_{\mathsf{BSC}}(m) \oplus e) = m.$$

3.2 Secret-Key and Public-Key Codes, With CPA and CCA Security

We begin by formalizing the notion of secret-key codes with CPA security, which will serve as the basic primitive in our constructions.

We use slightly different definitions and formulations than those in [AAC+25]. In particular, the notion of robustness defined in [AAC+25], termed *adaptive robustness*, implies our CPA notion of robustness defined below. In the secret-key setting these two notions are equivalent. However, this equivalence does not extend to the public-key setting (we elaborate on this after defining public-key codes).

Definition 3.5 (CPA Secret Key Codes)**.** A secret key (computational) code Π_{code} on message space $\{0,1\}^k$ and codeword space $\{0,1\}^n$ for $n = n(\lambda)$ is a set of efficient algorithms (Setup, Enc, Dec) with the following syntax:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{sk}$: On input security parameter λ, the setup algorithm outputs a secret key sk.
- $\mathsf{Enc}(\mathsf{sk}, x) \to c$: On input secret key sk and message $x \in \{0,1\}^k$, the encoding algorithm samples a codeword c.
- $\mathsf{Dec}(\mathsf{sk}, c') \to x$: On input secret key sk and string $c' \in \{0,1\}^n$, the decoding algorithm outputs a message x.

The rate of the code is defined to be $R = k/n$. If the scheme can only encode a singular message (i.e. $k = 0$), then we call it a **zero-bit** code,[8] and similarly, if it can encode one bit of information (i.e. $k = 1$), then we call it a **one-bit** code. We consider the following properties for Π_{code}:

- **CPA p-Robustness:** For a security parameter $\lambda \in \mathbb{N}$, the CPA p-robustness $\mathsf{ExptAR}(\mathcal{A}, \lambda)$ for some $p = p(\lambda)$ between an efficient adversary $\mathcal{A}$ and challenger is as follows:
  1. On input security parameter 1^λ, the challenger samples $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda)$. The challenger prepares encoding oracle $O_E = \mathsf{Enc}(\mathsf{sk}, \cdot)$ for the adversary.
  2. The adversary $\mathcal{A}^{O_E}$ submits a challenge $x \in \{0,1\}^k$. The challenger samples $c \leftarrow \mathsf{Enc}(\mathsf{sk}, x)$ and sends c to $\mathcal{A}^{O_E}$.
  3. The adversary $\mathcal{A}^{O_E}$ submits a string c'. The output of the experiment is 1 iff $\delta(c', c) \leq p$ and $\mathsf{Dec}(\mathsf{sk}, c') \neq x$ (and 0 otherwise).

  We say a code Π_{PRC} is CPA p-robust if for all efficient adversaries $\mathcal{A}$, there exists $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$

$$\Pr[\mathsf{ExptAR}(\mathcal{A}, \lambda)] = \mathsf{negl}(\lambda).$$

- **CPA Pseudorandomness:** For a security parameter $\lambda \in \mathbb{N}$, the pseudorandomness experiment $\mathsf{ExptPR}^{(b)}(\mathcal{A}, \lambda)$ between an efficient adversary $\mathcal{A}$ and challenger is as follows:
  1. On input security parameter 1^λ, the challenger samples $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda)$. The challenger prepares encoding oracle $O_E = \mathsf{Enc}(\mathsf{sk}, \cdot)$ for the adversary.
  2. The adversary $\mathcal{A}^{O_E}$ submits a challenge $x \in \{0,1\}^k$. If $b = 0$, the challenger samples $c \leftarrow \mathsf{Enc}(\mathsf{sk}, x)$. If $b = 1$, the challenger samples $c \leftarrow \{0,1\}^n$. In either case, the challenger sends c to $\mathcal{A}^{O_E}$.
  3. The adversary $\mathcal{A}^{O_E}$ submits a guess bit b', which is the output of the experiment.

  We say a code Π_{code} is pseudorandom if for all efficient adversaries $\mathcal{A}$, there exists $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$

$$\left|\Pr[\mathsf{ExptPR}^{(0)}(\mathcal{A}, \lambda)] - \Pr[\mathsf{ExptPR}^{(1)}(\mathcal{A}, \lambda)]\right| = \mathsf{negl}(\lambda).$$

[8] To avoid trivial constructions, zero-bit codes must additionally satisfy a soundness property (see Section 6)

Public Key Codes and CPA p-Simulatable Decoding.

Definition 3.6 (Public Key Codes)**.** For (computational) codes in the public key setting, we adjust the syntax and security of the secret key codes as follows:

- The setup algorithm $\mathsf{Setup}(1^\lambda)$ outputs a public key pk in addition to sk.
- The encoding algorithm Enc takes in pk instead of sk.
- In step (1) of both the pseudorandomness and robustness games, the adversary additionally receives public key pk (rather than access to an encoding oracle).

Beyond the 'standard' definition of public key computational codes (with pseudorandomness and robustness) considered above, we also consider the following property. This will be needed to achieve CCA security in the public key setting.

Definition 3.7 (CPA p-Simulatable Decoding)**.** For a security parameter $\lambda \in \mathbb{N}$, the decoding simulation experiment $\mathsf{ExptDS}^{(b)}$ for some $p = p(\lambda)$ and simulator Sim between an efficient adversary and challenger is as follows:

1. On input security parameter 1^λ, the challenger samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$, and sends pk to the adversary.
2. As many times as desired:
   (a) The adversary $\mathcal{A}$ submits a challenge $x \in \{0,1\}^k$, encoding randomness r, and codeword c'. The challenger computes $c \leftarrow \mathsf{Enc}(\mathsf{pk}, x; r)$, checking that $\delta(c, c') \le p$ (aborting the experiment with output 0 otherwise).
   (b) If $b = 0$, the challenger outputs 1 iff $\mathsf{Dec}(\mathsf{sk}, c') = x$ and 0 otherwise.
   (c) If $b = 1$, the challenger returns $\mathsf{Sim}(\mathsf{pk}, x, r, c')$.
3. The adversary outputs a bit b', which is the output of the experiment.

We say a code Π_{code} satisfies p-simulatable decoding if there exists an efficient algorithm Sim such that for all efficient adversaries $\mathcal{A}$, there exists $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$

$$\left|\Pr[\mathsf{ExptDS}^{(0)}(\mathcal{A}, \lambda)] - \Pr[\mathsf{ExptDS}^{(1)}(\mathcal{A}, \lambda)]\right| = \mathsf{negl}(\lambda).$$

We note this is implied by the notion of public key adaptive robustness [AAC+25, Def. 3], which requires decoding succeed even when the adversary is allowed to specify both the message *and* encoding randomness (taking Sim as the always 1 function). Our CPA constructions will not realize adaptive robustness, but we will show in Section 7.3.1 that it can achieve simulation-decodability. Looking further ahead to Section 7.3, we can show this weaker property nevertheless suffices to argue security of a CPA to CCA transformation.

Remark 3.8. We note in our random oracle constructions Constructions 7.25 and 7.29, the simulator will also be able to observe the adversary's random oracle queries. This will only occur in the proofs of simulatable decoding/CCA security, and the corresponding definitions are otherwise unchanged.

Codes with CCA security.

Definition 3.9 (Chosen Codeword Attack (CCA) Security)**.** For both public and secret key codes, we also consider a stronger notion of CCA security - in other words security relative to a p-decoding oracle[9]. This modifies the security games as follows:

- In Step (1) of the robustness game, the challenger additionally prepares decoding oracle $O_D = \mathsf{Dec}(\mathsf{sk}, \cdot)$. which the adversary has access to for the remainder of the experiment.

[9]While syntactically this constant p could differ from the p in p-robustness, for our purposes we will only consider cases where these are equal.

- In Step (1) of the pseudorandomness game, the challenger once again prepares decoding oracle $\mathcal{O}_D = \mathsf{Dec}(\mathsf{sk}, \cdot)$ for the adversary, which the adversary has access to until it submits a challenge in Step (2). This is then replaced with a punctured decoding oracle

$$\mathcal{O}_{D^*}(c') = \begin{cases} x & \text{if } \delta(c, c') \leq p \\ \mathsf{Dec}(\mathsf{sk}, c') & \text{otherwise} \end{cases}$$

  which the adversary has access to for the remainder of the experiment.

Secret-key/public-key CPA/CCA list-decodable codes.

Definition 3.10 (List-Decodable Computational Codes)**.** Analogous to the traditional notion of list-decodable codes, we consider a relaxation of (both public and secret key) codes to the list decoding setting (enabling a larger decoding/robustness radius). This modifies the syntax and security of Definition 3.5 and Definition 3.9 as follows:

- There exists efficient algorithm $\mathsf{ListDec}(\mathsf{sk}, c') \to T$, which outputs a list T of messages.
- In the CPA p-robustness game, the output of the experiment is 1 iff $\delta(c', c) \leq p$ and $x \notin \mathsf{ListDec}(\mathsf{sk}, c')$.
- In the CPA p-simulatable decoding game, when $b = 0$, the challenger outputs 1 iff $x \in \mathsf{ListDec}(\mathsf{sk}, c')$ and 0 otherwise.
- When considering CCA security Definition 3.9, the decoding oracle and punctured decoding oracle are replaced with their list-decoding variants:

$$\mathcal{O}_D = \mathsf{ListDec}(\mathsf{sk}, \cdot) \quad , \quad \mathcal{O}_{D^*} = \begin{cases} \mathsf{ListDec}(\mathsf{sk}, c') \cup \{x\} & \text{if } \delta(c, c') \leq p \\ \mathsf{ListDec}(\mathsf{sk}, c') & \text{otherwise} \end{cases}$$

For both CPA security and CCA security, we say that the code is (p, L)-robust if in addition the robustness definition above it holds that size of the list is at most L.

3.2.1 Known Results for Pseudorandom Codes

We make use of the following prior constructions of pseudorandom codes.

Theorem 3.11 (Public-key CPA zero-bit codes, [AAC+25])**.** *Assuming* learning parity with noise (LPN) *with some constant noise rate is hard for all adversaries running in time* $2^{O(\sqrt{n})}$*:*

- *For every constant* $0 \leq p < 1/2$ *there exists a public-key CPA zero-bit code that is pseudorandom and* p*-robust.*
- *For every constant* $0 \leq p < 1/4$ *there exists a public-key CPA one-bit code that is pseudorandom and* p*-robust.*

Moreover, both codes satisfy p*-simulatable decoding.*

We note that the prior result of [AAC+25] does not state the simulatable decoding property explicitly, but their notion of adaptive-robustness implies it.

Theorem 3.12 (Secret-key CPA zero-bit codes, [CGG+26])**.** *Assuming the* "permuted codes" assumption [BHMW21, BW21, CGG+26] *the following holds:*

- *For every constant* $0 \leq p < 1/2$ *there exists a secret-key CPA zero-bit code that is pseudorandom and* p*-robust.*
- *For every constant* $0 \leq p < 1/4$ *there exists a secret-key CPA one-bit code that is pseudorandom and* p*-robust.*

The "permuted codes" assumption in its full generality says that some security property holds for any linear code with sufficiently large dual distance. To get the above, we need it to hold for some codes that are efficiently list-decodable for $p < 1/2$ errors or efficiently uniquely decodable for $p < 1/4$ errors respectively.

Upgrading the message domain size for secret-key CPA codes. The following result from [AAC+25] showed how to increase the domain size of the code, and preserve robustness. Here, we leave the rate unspecified and allow it to be arbitrarily poor, as long as the codeword length is polynomial size.

Theorem 3.13 (From one-bit codes to many-bit codes, [AAC+25])**.** *Let $0 < p, \varepsilon$ be constants such that $p + \varepsilon < 1/4$ and let $\lambda \in \mathbb{N}$ be the security parameter. Given a secret-key one-bit CPA code Π that is pseudorandom and $(p+\varepsilon)$-robust. Then there exists a CPA code $\Pi' = (\mathsf{Enc}, \mathsf{Dec})$ that is pseudorandom and p-robust such that $\Pi'.\mathsf{Enc} : \{0,1\}^{\lambda} \to \{0,1\}^{m(\lambda)}$ for $m \in \mathsf{poly}$.*

Upgrading CPA Secure Codes to CCA Secure Codes. We will make use of the following theorem from [AAC+25] that showed how to upgrade a secret key CPA pseudorandom code into a secret key CCA pseudorandom code, we will make use of this transformation.

Theorem 3.14 (From secret key CPA codes to CCA codes, [AAC+25])**.** *Given a secret-key CPA code Π that is pseudorandom and p-robust with rate R. Then for any constant $0 < \varepsilon$, there exists a secret-key CCA code Π' that is pseudorandom and p-robust with rate $R' = R - \varepsilon$.*

3.3 Correlation-Intractable Hashing

We recall the definition of 2-input correlation-intractable hashing against shifted output relations from [LV22, LSW25], which will be used in Section 7.2 to achieve unique decoding in the plain model.

Notation. For a matrix $\mathbf{M} \in \mathbb{F}_2^{r \times n}$ and a subset $S \subseteq [n]$, we write $[\mathbf{M}]_S \in \mathbb{F}_2^{r \times |S|}$ for the submatrix of $\mathbf{M}$ formed by the columns indexed by S. We write $\mathsf{Perm}(n)$ for the symmetric group on $[n]$.

Definition 3.15 (Shifted Output Relation [LV22])**.** A *shifted output relation* is defined by a tuple (R_{out}, f), where $R_{\mathsf{out}} \subseteq (\{0,1\}^v)^2$ is an output relation and $f : \{0,1\}^k \to \{0,1\}^v$ is a function. The induced 2-ary relation $R = (R_{\mathsf{out}}, f)$ is defined as:

$$R\big((x,y),(x',y')\big) \;\Leftrightarrow\; \big(y \oplus f(x),\; y' \oplus f(x')\big) \in R_{\mathsf{out}} \;\wedge\; x \neq x'.$$

We say $R = (R_{\mathsf{out}}, f)$ is s**-sparse** if $|R_{\mathsf{out}}|/2^{2v} \leq 2^{-s}$. Sparsity is defined only with respect to R_{out} and not f.

Definition 3.16 (Efficiently Checkable Relation Ensemble)**.** A relation ensemble $\mathcal{R} = \{R_\lambda = (R_{\mathsf{out},\lambda}, f_\lambda)\}_{\lambda \in \mathbb{N}}$ of shifted output relations is **efficiently checkable** if there exists a polynomial poly such that for all $\lambda \in \mathbb{N}$, both f_λ and $R_{\mathsf{out},\lambda}$ are computable in $\mathsf{poly}(\lambda)$ time.

Definition 3.17 (2-Input Correlation-Intractable Hashing [LV22])**.** Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ be a function family ensemble and $s = s(\lambda, k, v)$. A keyed hash function $\Pi_{\mathsf{Hash}} = (\mathsf{Gen}, \mathsf{Hash})$ with $\mathsf{Gen} : 1^\lambda \times 1^k \times 1^v \to \mathcal{K}$ and $\mathsf{Hash} : \mathcal{K} \times \{0,1\}^k \to \{0,1\}^v$ is **2-input correlation intractable (2-MICIH) for $\mathcal{F}$-shifted s-sparse output relations** if for every efficiently checkable 2-ary shifted output relation ensemble $\mathcal{R} = \{R_\lambda = (R_{\mathsf{out},\lambda}, f_\lambda)\}$ satisfying:

- $f_\lambda \in \mathcal{F}_\lambda$ for all $\lambda \in \mathbb{N}$, and
- R_λ is $s(\lambda, k, v)$-sparse for all $\lambda \in \mathbb{N}$,

and for all polynomial-time adversaries $\mathcal{A}$, there exists a negligible function negl such that

$$\Pr\Big[K \leftarrow \mathsf{Gen}(1^\lambda, 1^k, 1^v);\; (x,x') \leftarrow \mathcal{A}(K) :\; R_\lambda\big((x, \mathsf{Hash}(K,x)),\; (x', \mathsf{Hash}(K,x'))\big) = 1\Big] = \mathsf{negl}(\lambda).$$

When $\mathcal{F}$ is the family of all $\mathbb{F}_2$-linear functions $\{0,1\}^k \to \{0,1\}^v$, we call Π_{Hash} a **linearly shifted 2-MICIH**.

Theorem 3.18 ([LSW25])**.** *Assuming any of* $\{\mathsf{DDH}, \mathsf{LWE}, \mathsf{QR}, \mathsf{DCR}\}$*, there exists a polynomial $s_0 = s_0(\lambda_{\mathsf{Hash}})$ and a linearly shifted 2-MICIH for s_0-sparse output relations (with input length k and output length v for any $k, v = \mathsf{poly}(\lambda_{\mathsf{Hash}})$). Moreover, assuming any of these with subexponential security 2^{λ^c}, the resulting hash achieves subexponential CI security $2^{\lambda_{\mathsf{Hash}}^{c'}}$ for some constant $c' > 0$.*

We emphasize that the sparsity threshold $s_0(\lambda_{\mathsf{Hash}})$ is a fixed polynomial determined by the underlying assumption and construction, and does not depend on the input or output lengths k, v.

4 Shannon-Rate Secret Key Codes

In this section we show a simple transformation that takes a one-bit CPA secret-key pseudorandom code with $p < 1/4$-robustness, and an arbitrarily poor rate, and outputs a CCA secret-key code for arbitrary length messages with p-robustness and the optimal Shannon rate $1 - H(p) - \varepsilon$.

Theorem 4.1. *Let $p, \varepsilon > 0$ be constants such that $p < 1/4$. Suppose there exists a* one-bit *secret-key CPA code that is pseudorandom and $(p + \varepsilon)$-robust. Then there exists a secret-key CCA code that is pseudorandom and p-robust for arbitrary length messges with rate $R = 1 - H(p) - \varepsilon$.*

Secret-key one-bit CPA pseudorandom codes with constant robustness can be obtained from several known constructions. In particular, such codes are known to exist assuming sub-exponential *learning parity with noise (LPN)* assumption [CG24, AAC+25], or a *permuted codes assumption* (for a code with error tolerance p) introduced in [CGG+26]. Applying Theorem 4.1 to any such construction immediately yields secret-key CCA-secure pseudorandom codes that are p-robust and achieve rate $1 - H(p) - \varepsilon$, matching the Shannon capacity up to an arbitrarily small loss.

Construction. The proof for the above theorem is given below. Note that by Theorem 3.14 any CPA secure pseudorandom code can be generically transformed into a CCA secure pseudorandom code with essentially the same rate. Thus, in the following construction we start by presenting a CPA secure code with a rate that matches the Shannon bound.

Construction 4.2. Let $\lambda \in \mathbb{N}$ be a security parameter and let $0 < p, \alpha$ be constants such that $p + \alpha < 1/4$, and let $m, n \in \mathsf{poly}(\lambda)$. We now describe the ingredients:

- Let $\Pi_{\mathsf{PRC}} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ be a $(p + \alpha)$-robust secret key pseudorandom code from $\{0,1\}^\lambda$ to $\{0,1\}^m$.
- Let $\Pi_{\mathsf{BSC}} = (\mathsf{Enc}, \mathsf{Dec})$ be a code from $\{0,1\}^k$ to $\{0,1\}^n$ that decodes from permuted errors of relative weight $p + \alpha$ (see Theorem 3.4).
- Let PRG be a pseudorandom generator with seed length λ and output in $\{0,1\}^n$.

We now describe the new code $\Pi = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{sk}$: On input security parameter λ, the setup algorithm simply runs $\mathsf{sk}_{\mathsf{PRC}} \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{Setup}(1^\lambda)$ and samples two random permutations $\pi : [n+m] \to [n+m]$, and $\pi_d : [n] \to [n]$ and outputs $\mathsf{sk} = (\mathsf{sk}_{\mathsf{PRC}}, \pi, \pi_d)$.
- $\mathsf{Enc}(\mathsf{sk}, x) \to c$: On input secret key $\mathsf{sk} = (\mathsf{sk}_{\mathsf{PRC}}, \pi, \pi_d)$ and message $x \in \{0,1\}^k$, the encoding algorithms computes the following:
  - Sample PRG seed $s \leftarrow \{0,1\}^\lambda$, and compute $w = \mathsf{PRG}(s)$.
  - Compute $c_{\mathsf{PRC}} \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{Enc}(s)$.
  - Compute encoding of the message $c_{\mathsf{BSC}} = \Pi_{\mathsf{BSC}}.\mathsf{Enc}(x)$.
  - Set $z = (w \oplus c_{\mathsf{BSC}})$, and $y = (v := \pi_d(z), c_{\mathsf{PRC}})$.
  - Output $c = \pi(y)$.
- $\mathsf{Dec}(\mathsf{sk}, c') \to x'$: On input secret key $\mathsf{sk} = (\mathsf{sk}_{\mathsf{PRC}}, \pi, \pi_d)$ and possibly noisy codeword c', the decoding algorithm computes the following:
  - Unpermute: $y' = \pi^{-1}(c')$, parse $y' = (v', c'_{\mathsf{PRC}})$ and compute $z' = \pi_d^{-1}(v')$.
  - Compute $s' = \Pi_{\mathsf{PRC}}.\mathsf{Dec}(c'_{\mathsf{PRC}})$.
  - Compute $w' = \mathsf{PRG}(s')$ and $c'_{\mathsf{BSC}} = w' \oplus z'$.
  - Output $x' = \Pi_{\mathsf{BSC}}.\mathsf{Dec}(c'_{\mathsf{BSC}})$.

The following is our technical lemma that argues the correctness of the above construction.

Lemma 4.3 (Robustness, pseudorandomness and rate for the construction). *Let $0 < p, \alpha$ be constants with $p + \alpha < 1/4$. Assume the ingredients in Construction 4.2. Then the code Π defined in Construction 4.2 is a CPA pseudorandom and p-robust with rate $R = k/(n+m)$.*

The proof for Lemma 4.3 is given in Section 4.1, but first we prove Theorem 4.1.

Proof of Theorem 4.1. For constants $0 < p, \varepsilon$ such that $p < 1/4$. By assumption we are given a one-bit CPA secret-key pseudorandom code. For any constant $\varepsilon' < \varepsilon$ applying Theorem 3.13, we obtain a secret-key CPA pseudorandom code Π_{PRC} from λ bits to $m(\lambda)$ bits that is $(p + \varepsilon')$-robust.

Let $0 < \alpha \leq \varepsilon'$ be a sufficiently small constant such that

$$1 - H(p + \alpha) - \alpha \geq 1 - H(p) - \varepsilon/2.$$

Note that since $\alpha < \varepsilon'$, Π_{PRC} is also $(p + \alpha)$-robust. Let Π_{BSC} be the code guaranteed in Theorem 3.4 from k bits to n bits with rate $1 - H(p + \alpha) - \alpha$, and let k and n to be sufficiently large such that $m < n \cdot (\varepsilon/3)$. Let Π be the code defined in Construction 4.2 using ingredient Π_{PRC}, Π_{BSC} and a pseudorandom generator PRG (which is implied by one way functions and hence Π_{PRC}). By Lemma 4.3 it follows that the code Π is CPA p-robust and pseudorandom with rate

$$R = \frac{k}{n+m} \geq \frac{k}{n \cdot (1 + \varepsilon/3)} \geq \frac{k}{n} \cdot (1 - \varepsilon/3) \geq (1 - H(p) - \varepsilon/2) \cdot (1 - \varepsilon/3) \geq 1 - H(p) - \varepsilon \cdot 5/6$$

Finally, by Theorem 3.14 it follows that there exists a code Π' that is CCA p-robust and pseudorandom with rate $1 - H(p) - \varepsilon$ as desired. □

4.1 Proof of Lemma 4.3

We make use of the following notation. In Construction 4.2, the secret key is $\mathsf{sk} = (\mathsf{sk}_{\mathsf{PRC}}, \pi, \pi_d)$ where π_d and π are independent uniformly random permutations, where π_d is a permutation on $[n]$ and π is permutation on $[N]$ where $N \coloneqq n + m$ is the codeword size of Π.

Rate. It follows immediately by construction that the code has rate $R = k/N = k/(n+m)$ as desired.

The code is pseudorandom. We now prove the pseudorandom property of the code. In fact, we prove something mildly stronger which we will make use of when we prove the robustness property. For a security parameter $\lambda \in \mathbb{N}$ and an efficient adversary $\mathcal{A}$, we define the follows experiment:

$\mathsf{StrongExptPR}^{(b)}(\mathcal{A}, \lambda)$

1. On input security parameter 1^λ, the challenger samples $\mathsf{sk} = (\mathsf{sk}_{\mathsf{PRC}}, \pi, \pi_d) \leftarrow \mathsf{Setup}(1^\lambda)$. The challenger prepares encoding oracle $O_E = \Pi.\mathsf{Enc}(\mathsf{sk}, \cdot)$ for the adversary.
2. The adversary $\mathcal{A}^{O_E}$ submits a challenge $x \in \{0,1\}^k$. If $b = 0$, the challenger samples $c \leftarrow \Pi.\mathsf{Enc}(\mathsf{sk}, x)$. If $b = 1$, the challenger samples $c \leftarrow \{0,1\}^n$. In either case, the challenger sends c with π and π_d to $\mathcal{A}^{O_E}$.
3. The adversary $\mathcal{A}^{O_E}$ submits a guess bit b', which is the output of the experiment.

We prove that

Claim 4.4. $\left|\Pr[\mathsf{StrongExptPR}^{(0)}(\mathcal{A}, \lambda)] - \Pr[\mathsf{StrongExptPR}^{(1)}(\mathcal{A}, \lambda)]\right| \leq \mathsf{negl}(\lambda)$

Note that this implies the pseudorandom property since

$$\left|\Pr[\mathsf{ExptPR}^{(0)}(\mathcal{A}, \lambda)] - \Pr[\mathsf{ExptPR}^{(1)}(\mathcal{A}, \lambda)]\right| \leq \left|\Pr[\mathsf{StrongExptPR}^{(0)}(\mathcal{A}, \lambda)] - \Pr[\mathsf{StrongExptPR}^{(1)}(\mathcal{A}, \lambda)]\right|.$$

Where $\mathsf{ExptPR}^{(b)}(\mathcal{A}, \lambda)$ is the standard pseudorandom experiment defined in Definition 3.5. That is, StrongExptPR implies that the pseudorandom property should hold even when the adversary is given access to π and π_d as an input.

Proof of Claim 4.4. Consider the following simple hybrid

H_0: Same as $\mathsf{StrongExptPR}^{(0)}(\mathcal{A}, \lambda)$, except replace $c_{\mathsf{PRC}} = \Pi_{\mathsf{PRC}}.\mathsf{Enc}(s)$ by a truly uniform string $u \leftarrow \{0,1\}^m$ in every encoding query. By (a straightforward hybrid over) pseudorandomness of Π_{PRC}, and since $\mathsf{PRG}(s)$ is efficiently computable given s,

$$\left|\Pr[\mathsf{StrongExptPR}^{(0)}(\mathcal{A}, \lambda)] - \Pr[H_0]\right| \leq \mathsf{negl}(\lambda).$$

H_1: Same as H_0, except replace $\mathsf{PRG}(s)$ by a truly uniform $w \leftarrow \{0,1\}^n$ and note that this means that H_1 is identical to $\mathsf{StrongExptPR}^{(1)}(\mathcal{A}, \lambda)$. By the security of the PRG,

$$\left|\Pr[H_0] - \Pr[H_1]\right| \leq \mathsf{negl}(\lambda).$$

The above implies that

$$\left|\Pr[\mathsf{StrongExptPR}^{(0)}(\mathcal{A}, \lambda)] - \Pr[\mathsf{StrongExptPR}^{(1)}(\mathcal{A}, \lambda)]\right| = \mathsf{negl}(\lambda),$$

which concludes the proof. □

The code is p-robust.

For fixed security parameter $\lambda \in \mathbb{N}$ and efficient adversary $\mathcal{A}$ let $\mathsf{ExptAR} \coloneqq \mathsf{ExptAR}(\mathcal{A}, \lambda)$ denote the robustness experiment as defined in Definition 3.5. That is,

ExptAR

1. On input security parameter 1^λ, the challenger samples $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda)$. The challenger prepares encoding oracle $O_E = \mathsf{Enc}(\mathsf{sk}, \cdot)$ for the adversary.
2. The adversary $\mathcal{A}^{O_E}$ submits a challenge $x \in \{0,1\}^k$. The challenger samples $c \leftarrow \mathsf{Enc}(\mathsf{sk}, x)$ and sends c to $\mathcal{A}^{O_E}$.
3. The adversary $\mathcal{A}^{O_E}$ submits a string c'. The output of the experiment is 1 iff $\delta(c', c) \leq p$ and $\mathsf{Dec}(\mathsf{sk}, c') \neq x$ (and 0 otherwise).

We now define an alternative decoding experiment that depends only on the error pattern induced and the permutations.

Alternative decoding predicate. For a message x, permutations π, π_d, and an error pattern e, define $f(x, \pi, \pi_d, e) = 1$ if and only if the following conditions hold for $e^\pi \coloneqq \pi^{-1}(e) = (e_d, e_c) \in \{0,1\}^n \times \{0,1\}^m$:

1. $\delta(e_d) \leq p + \alpha$ and $\delta(e_c) \leq p + \alpha$, and
2. $\Pi_{\mathsf{BSC}}.\mathsf{Dec}\left(\Pi_{\mathsf{BSC}}.\mathsf{Enc}(x) \oplus \pi_d^{-1}(e_d)\right) = x$.

Otherwise, define $f(x, \pi, \pi_d, e) = 0$.

We now define a new decoding experiment Exp^f that uses the predicate f. Specifically,

Exp^f:

1. On input security parameter 1^λ, the challenger samples $\mathsf{sk} = (\mathsf{sk}_{\mathsf{PRC}}, \pi, \pi_d) \leftarrow \mathsf{Setup}(1^\lambda)$. The challenger prepares encoding oracle $O_E = \mathsf{Enc}(\mathsf{sk}, \cdot)$ for the adversary.
2. The adversary $\mathcal{A}^{O_E}$ submits a challenge $x \in \{0,1\}^k$. The challenger samples $c \leftarrow \mathsf{Enc}(\mathsf{sk}, x)$ and sends c to $\mathcal{A}^{O_E}$.
3. The adversary $\mathcal{A}^{O_E}$ submits a string c'. The output of the experiment is 1 iff $\delta(c', c) \leq p$ and

$$f(x, \pi, \pi_d, e \coloneqq c \oplus c') \neq 1.$$

Finally, consider the experiment $\mathsf{Exp}^{f,ps}$: Define identically to Exp^{f}, but where the oracle O_E returns random strings (instead of codewords) and the challenge c is uniformly sampled. That is,

$\mathsf{Exp}^{f,ps}$:

1. On input security parameter 1^λ, the challenger samples $\mathsf{sk} = (\mathsf{sk}_{\mathsf{PRC}}, \pi, \pi_d) \leftarrow \mathsf{Setup}(1^\lambda)$. The challenger prepares encoding oracle O_E that when called returns uniform random strings of size N.
2. The adversary $\mathcal{A}^{O_E}$ submits a challenge $x \in \{0,1\}^k$. The challenger samples $c \leftarrow \{0,1\}^N$ and sends c to $\mathcal{A}^{O_E}$.
3. The adversary $\mathcal{A}^{O_E}$ submits a string c'. The output of the experiment is 1 iff $\delta(c', c) \le p$ and

$$f(x, \pi, \pi_d, e \coloneqq c \oplus c') \neq 1.$$

The proof for robustness is immediate from the following claims:

Claim 4.5. $\Pr[\mathsf{Exp}^{f}] \ge \Pr[\mathsf{ExptAR}] - \mathsf{negl}(\lambda)$.

Claim 4.6. $\left|\Pr[\mathsf{Exp}^{f}] - \Pr[\mathsf{Exp}^{f,ps}]\right| \le \mathsf{negl}(\lambda)$.

Claim 4.7. $\Pr[\mathsf{Exp}^{f,ps}] \le \mathsf{negl}(\lambda)$.

Putting together Claims 4.5 to 4.7 we conclude that $\Pr[\mathsf{ExptAR}] = \mathsf{negl}$ as desired. Thus it suffices to prove Claims 4.5 to 4.7 which we do below.

Proof of Claim 4.5. The experiments ExptAR and Exp^{f} are identical except for the final predicate. In both experiments, the challenger samples x, computes $c \leftarrow \mathsf{Enc}(\mathsf{sk}, x)$, the adversary outputs c', and we define the induced error pattern

$$e \coloneqq c \oplus c'.$$

Conditioned on the event $\delta(c', c) \le p$, we have $\mathsf{ExptAR} = 1$ iff $\mathsf{Dec}(\mathsf{sk}, c') \neq x$, whereas $\mathsf{Exp}^{f} = 1$ iff $f(x, \pi, \pi_d, e) \neq 1$. Hence it suffices to show that

$$f(x, \pi, \pi_d, e) = 1 \quad \Longrightarrow \quad \mathsf{Dec}(\mathsf{sk}, c') = x$$

except with negligible probability.

Assume $f(x, \pi, \pi_d, e) = 1$. By definition of f, letting $e^\pi \coloneqq \pi^{-1}(e) = (e_d, e_c)$ we have $\delta(e_c) < p + \alpha$, $\delta(e_d) < p + \alpha$ and

$$\Pi_{\mathsf{BSC}}.\mathsf{Dec}\big(\Pi_{\mathsf{BSC}}.\mathsf{Enc}(x) \oplus \pi_d^{-1}(e_d)\big) = x.$$

Now consider $\mathsf{Dec}(\mathsf{sk}, c')$. The decoder first unpermutes

$$\pi^{-1}(c') = \pi^{-1}(c \oplus e) = \pi^{-1}(c) \oplus e^\pi.$$

Writing $\pi^{-1}(c) = y = (v, c_{\mathsf{PRC}})$, the PRC component is corrupted by e_c, and when after inverting π_d, that is $z = \pi_d^{-1}(v)$, the data component is corrupted with $\pi_d^{-1}(e_d)$. By the $(p+\alpha)$-robustness of Π_{PRC} and the fact that $\delta(e_c) < p+\alpha$, the decoder recovers the correct PRG seed s except with negligible probability, and therefore reconstructs the correct mask $w = \mathsf{PRG}(s)$. Removing the mask yields the codeword $\Pi_{\mathsf{BSC}}.\mathsf{Enc}(x)$ whose corruption is precisely $\pi_d^{-1}(e_d)$, and so by the assumption that $f(x, \pi, \pi_d, e) = 1$ the final decoding outputs x. This proves the claim. □

Proof of Claim 4.6. The claim follows directly from the strong pseudorandomness property established in Claim 4.4. In particular, the pseudorandomness guarantee holds even when the adversary is given the permutations π and π_d. Moreover, by a simple hybrid argument we can assume that all calls made to the oracle $\mathcal{A}^{O_E}$ in the experiment $\mathsf{StrongExptPR}^{(1)}(\mathcal{A}, \lambda)$ return a random string.

Observe that in both experiments Exp^{f} and $\mathsf{Exp}^{f,ps}$ the winning condition is identical, namely $f(x, \pi, \pi_d, e) \neq 1$, where $e = c \oplus c'$. Moreover, this predicate depends only on (x, π, π_d, e) (and does not require the knowledge of the PRC seed).

Therefore, by the strong pseudorandomness property, the joint distributions of (x, π, π_d, e) in the two experiments are computationally indistinguishable, and hence

$$\left|\Pr[\mathsf{Exp}^{f} = 1] - \Pr[\mathsf{Exp}^{f,ps} = 1]\right| \leq \mathsf{negl}(\lambda),$$

which proves the claim. □

Proof of Claim 4.7. Fix any PPT adversary $\mathcal{A}$ and consider the experiment $\mathsf{Exp}^{f,ps}$. Let $c \in \{0,1\}^N$ be the (uniform) challenge string and let c' be the adversary's output. Define the induced error vector

$$e \coloneqq c \oplus c'.$$

Notice that in the experiment $\mathsf{Exp}^{f,ps}$ the permutations π and π_d are random and independent from e and c. Let $e^{\pi} \coloneqq \pi^{-1}(e) = (e_d, e_c)$, and recall that $\mathsf{Exp}^{f,ps} = 1$ iff $\mathsf{wt}(e) \leq pN$ and $f(x, \pi, \pi_d, e) \neq 1$.

Let $T \subseteq [N]$ denote the (hidden) set of positions that correspond, under π^{-1}, to the PRC block, so $|T| = m$. Since c is uniform and independent of π, the adversary's (possibly randomized) choice of E as a function of c is independent of the random subset T. Therefore $\mathbb{E}[\mathsf{wt}(e_c)] \leq p \cdot |T| \leq pm$, and by standard tail bound for hypergeometric sampling (e.g., Hoeffding),

$$\Pr\left[\mathsf{wt}(e_c) > (p+\alpha)m\right] \leq 2^{-\Omega(\alpha^2 m)}.$$

and similarly, $\Pr\left[\mathsf{wt}(e_d) > (p+\alpha)n\right] \leq 2^{-\Omega(\alpha^2 n)}$. Define the good splitting event

$$\mathsf{GoodSplit} \coloneqq \left\{\mathsf{wt}(e_c) \leq (p+\alpha)m \ \wedge \ \mathsf{wt}(e_d) \leq (p+\alpha)n\right\}.$$

By a union bound, $\Pr[\neg\mathsf{GoodSplit}] = \mathsf{negl}(\lambda)$ (here we implicitly assume that m, n are $\Omega(\lambda)$). By the properties of the code Π_{BSC} (see, Theorem 3.4), it holds if $\mathsf{wt}(e_d) \leq (p+\alpha) \cdot n$,

$$\Pr[\Pi_{\mathsf{BSC}}.\mathsf{Dec}\big(\Pi_{\mathsf{BSC}}.\mathsf{Enc}(x) \oplus \pi_d^{-1}(e_d)\big) = x \wedge \mathsf{GoodSplit}] \geq 1 - \mathsf{negl}(\lambda).$$

Since $f(x, \pi, \pi_d, e) = 1$ iff $\{\Pi_{\mathsf{BSC}}.\mathsf{Dec}\big(\Pi_{\mathsf{BSC}}.\mathsf{Enc}(x) \oplus \pi_d^{-1}(e_d)\big) = x \wedge \mathsf{GoodSplit}\}$, it follows that $\Pr[\mathsf{Exp}^{f,ps} = 1] = \mathsf{negl}(\lambda)$, which proves the claim. □

5 Heuristic Seeded Codes via Ideal Obfuscation

We observe that the CCA-secure secret-key codes from the previous section can be transformed into seeded codes by obfuscating the encoding and decoding procedures and publishing the resulting obfuscated programs as the seed. In more detail, we can first make our CCA-secure secret key codes deterministic by using a pseudorandom function (PRF) of the message as the encoding randomness. The resulting code no longer satisfies the definition of pseudorandomness (which inherently requires a randomized code), but it is easy to see that it maintains robustness. We then take the resulting deterministic encoding and decoding procedures with a hard-coded secret key, obfuscate them, and publish the obfuscated programs as the seed. In the ideal obfuscation model, the adversary only gets black-box access to the obfuscated program, which is the same as oracle access to the encoding and decoding procedures. Therefore, the CCA security of the secret key codes ensures the security of the seeded code in the ideal obfuscation model.

Theorem 5.1. *Let $p, \varepsilon > 0$ be constants such that $p < 1/4$. Suppose there exists a* one-bit *secret-key CPA code that is pseudorandom and $(p+\varepsilon)$-robust. Then, in the idea obfuscation model, there exists a seeded code that is p-robust for arbitrary length messages with rate $R = 1 - H(p) - \varepsilon$.*

While ideal obfuscation does not exist, constructions in the ideal obfuscation model give heuristic evidence that security holds when the idea obfuscation is replaced with indistinguishability obfuscation, analogously to random oracle constructions giving heuristic evidence of security when replaced with a real-world hash functions. In fact, this analogy was formalized in the work of [JLLW23], which showed that one can instantiate ideal obfuscation in a variant of the random oracle model called the *pseudorandom oracle model* (PRO), assuming indistinguishability obfuscation. Combining the above, we get a construction of seeded codes that achieve Shannon capacity in the PRO model assuming secret-key PRCs and indistinguishability obfuscation. Overall, this yields the first heuristic candidate for seeded codes achieving Shannon capacity for $p < 1/4$.

6 List Decodable Pseudorandom Codes With Optimal Robustness

In this section we present a transformation from zero-bit pseudorandom codes to list-decodable codes with large message spaces and optimal robustness. In the list-decoding setting, our constructions tolerate error rates throughout the full range $0 \leq p < \frac{1}{2}$. This construction works for both secret-key and public-key PRCs, but we mainly focus on the latter.

Zero-Bit Pseudorandom Codes and Soundness. Our construction here will also make use of zero-bit pseudorandom codes, which, in addition to satisfying, the previous notions of pseudorandomness and p-robustness, will also need to satisfy *soundness*.

- **Soundness:** For a security parameter $\lambda \in \mathbb{N}$, a computational code $\Pi_{\mathsf{PRC}} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ is *sound* if for all fixed bit-strings $c' \in \{0,1\}^n$:

$$\Pr[\mathsf{Dec}(\mathsf{sk}, c) = \perp \mid (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)] = 1 - \mathsf{negl}(\lambda),$$

  where the probability is taken over the randomness of λ.

Soundness for secret-key codes is defined analogously. As noted in [CG24], this condition is needed in the zero-bit case to ensure a meaningful primitive (otherwise, a trivial 'pseudorandom code' could be constructed by considering all strings in $\{0,1\}^n$ as codewords). As such, all constructions of (zero-bit) pseudorandom codes satisfy soundness.

We now state our result for list-decodable codes with (p, L)-robustness for $0 \leq p < 1/2$. To the best of our knowledge the notion of list-decoding for pseudorandom codes was not formalized and explored prior to this work. The results below are stated for public-key codes, but the construction and proof also hold for the secret-key codes.

Theorem 6.1 (List-decodable pseudorandom: From zero-bit to large message domain)**.** *Let $p \geq 0, \varepsilon > 0$ be constants such that $p + \varepsilon < \frac{1}{2}$. Suppose there exists a public-key [resp., secret-key] CPA zero-bit pseudorandom code that is $(p+\varepsilon)$-robust and sound. Then there exists polynomials $n(\lambda)$ and $L(\lambda)$ and an explicit public-key [resp., secret-key] CPA list-decodable code $\Pi = (\mathsf{Enc}, \mathsf{Dec}_{\mathsf{ListDec}})$ with $\mathsf{Enc} : \{0,1\}^\lambda \to \{0,1\}^{n(\lambda)}$, such that Π is pseudorandom and $(p, L(\lambda))$-robust. Moreover, if the zero-bit code has $(p+\varepsilon)$-simulatable decoding then Π also has p-simulatable decoding.*

Note that the result of [AAC+25] for zero-bit codes stated in Theorem 3.11 satisfy the simulatable decoding property, and suffice to instantiate this theorem from LPN in the public (or secret) key setting, and the result of [CGG+26] suffices to instantiate this theorem in the secret key setting from permuted codes.

The proof for Theorem 6.1 is given in section Section 6.3.

6.1 Polynomial Message Space List-Decodable Pseudorandom Codes

We begin by showing how to obtain list-decodable pseudorandom codes for any polynomial-size message space, assuming the existence of a zero-bit pseudorandom code. The construction is conceptually simple: we associate each possible message with an independent instance of the underlying zero-bit code and decode by collecting all indices whose corresponding code accepts. The main technical challenge is to argue that the resulting list size remains polynomial in the security parameter and, crucially, independent of the message space size. The following lemma formalizes this guarantee.

Lemma 6.2. *Let $p < \frac{1}{2}$ be an arbitrary constant and let $\lambda \in \mathbb{N}$ be a security parameter. Suppose there exists a public-key [resp., secret-key] CPA zero-bit pseudorandom code which is sound and p-robust. Then there exists a polynomial $L(\lambda)$ such that, for any polynomial $q(\lambda)$, there is a public-key [resp., secret-key] CPA list-decodable code Π with message space $[q]$ that is pseudorandom and $(p, L(\lambda))$-robust. Importantly, note that L is a fixed function of the security parameter and independent of q. Moreover, if the zero-bit code has p-simulatable decoding then Π also has p-simulatable decoding.*

Proof of Lemma 6.2. We make use of the following construction:

Construction 6.3. Take the message space q to be an arbitrary polynomial in λ.

- Let $\Pi_{\mathsf{PRC}} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ be a zero-bit public-key pseudorandom code with codeword space $\{0,1\}^m$.

Our construction proceeds as follows:

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: On input security parameter λ, the setup algorithm samples an independent key pair $(\mathsf{pk}^{(i)}, \mathsf{sk}^{(i)}) \leftarrow \mathsf{Setup}(1^\lambda)$ for each $i \in [q]$. It returns $\mathsf{pk} = \{\mathsf{pk}^{(i)}\}_{i\in[q]}, \mathsf{sk} = \{\mathsf{sk}^{(i)}\}_{i\in[q]}$.
- $\mathsf{Enc}(\mathsf{pk}, x) \to c$: On input public key $\mathsf{pk} = \{\mathsf{pk}^{(i)}\}_{i\in[q]}$, message $x \in [q]$, the encoding algorithm samples and returns codeword $c \leftarrow \mathsf{Enc}(\mathsf{pk}^{(x)}, 1)$.
- $\mathsf{ListDec}(\mathsf{sk}, c') \to L$: On input secret key $\mathsf{sk} = \{\mathsf{sk}^{(i)}\}_{i\in[q]}$ and string $c' \in \{0,1\}^m$, the decoding algorithm computes and outputs set
$$L = \{i \in [q] \mid \mathsf{Dec}(\mathsf{sk}^{(i)}, c') = 1\}.$$

Since codewords of Construction 6.3 are simply codewords of the underlying Π_{PRC}, pseudorandomness of this construction follows immediately from pseudorandomness of Π_{PRC}. Similarly, p-robustness follows from p-robustness of Π_{PRC} (recall in the list-decoding robustness game Definition 3.10, it suffices for the codeword x to be contained in the output list L, which corresponds exactly to a noisy codeword of the selected zero-bit code decoding to 1). In the same vein, we can satisfy simulated decoding Construction 6.3 by defining simulator $\mathsf{Sim}(\mathsf{pk}, x, r, c)$ to simply invoke the underlying simulator $\Pi_{\mathsf{PRC}}.\mathsf{Sim}(\mathsf{pk}^{(x)}, 1, r, c)$. Thus, to complete Lemma 6.2, it remains to argue a bound on the list size of Construction 6.3.

Claim 6.4. *With overwhelming probability (over the randomness of* Setup*), the list L output by* ListDec *is of size $< m$ on all strings $\in \{0,1\}^m$.*

Proof. Fix an arbitrary string $c^* \in \{0,1\}^m$. Recall that by soundness of Π_{PRC}, the probability $\mathsf{Dec}(\mathsf{sk}, c^*) = 1$ is negligible over the randomness of sampling sk in Setup. Let $f = f(\lambda)$ refer to this negligible function. Thus, in our scheme for any fixed subset $T \subset [q]$ of m keys, the probability all of $\{\mathsf{sk}^{(i)}\}_{i\in T}$ decode c^* to 1 can be bounded by f^m, as the keys are sampled independently.

We can now union bound over all 2^m strings c^* and all $\binom{q}{m} \le q^m$ size m subsets of $[q]$ to conclude that the probability there exists *any* string c^* and size m subset of keys $\{\mathsf{sk}^{(i)}\}_{i\in T}$ which decode c^* to 1 can be bounded by $f^m \cdot q^m \cdot 2^m = (2 \cdot q \cdot f)^m$. Since q is polynomially bounded while f is negligible, $2 \cdot q \cdot f$ is negligible as well. □

Notice that m is simply a parameter Π_{PRC}, which is instantiated independently of q, and so this is sufficient to complete the proof of our lemma. □

6.2 From Polynomial Message Space to Exponential Message Space

We now extend the previous construction to support an exponential-size message space by combining pseudorandom list-decodable codes with standard list-recoverable codes via concatenation. The main idea is to encode the message using a list-recoverable outer code and then protect each outer symbol using an independent instance of the underlying pseudorandom code. The resulting scheme preserves pseudorandomness while tolerating error rates up to $p < \frac{1}{2}$. Crucially, this transformation preserves the list size up to a polynomial factor, so the final list size remains of the same order as that of the underlying pseudorandom code.

Construction 6.5. Let $0 < p, \varepsilon$ be constants such that $0 \le p + \varepsilon < 1/2$. Ingredients:

- Let $q \in \mathsf{poly}$ and let $\Pi_{\mathsf{PRC}} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{ListDec})$ be a $(p + \varepsilon, \ell_{\mathsf{PRC}})$-robust list-decodable pseudorandom code on message space $[q(\lambda)]$ where λ is the security parameter.
- Let $n, k \in \mathsf{poly}$, and let $\Pi_{\mathsf{LRC}} = (\mathsf{Enc}, \mathsf{Dec})$ be an (ε, ℓ, L)-list-recoverable code such that $\mathsf{Enc} : \{0,1\}^{k(\lambda)} \to [q(\lambda)]^{n(\lambda)}$, and $\ell = \ell_{\mathsf{PRC}}$.

Our construction proceeds as follows:

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: On input security parameter λ, the setup algorithm simply runs the $(\mathsf{pk}_{\mathsf{PRC}}, \mathsf{sk}_{\mathsf{PRC}}) \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{Setup}(1^\lambda)$ and outputs $\mathsf{pk} = \mathsf{pk}_{\mathsf{PRC}}, \mathsf{sk} = \mathsf{sk}_{\mathsf{PRC}}$.
- $\mathsf{Enc}(\mathsf{pk}, x) \to c$: On input public key $\mathsf{pk} = \mathsf{pk}_{\mathsf{PRC}}$ and input x, the encoding algorithm computes the following:
  - Compute $a_1, \ldots a_n \leftarrow \Pi_{\mathsf{LRC}}.\mathsf{Enc}(x)$.
  - For each $i \in [n]$, compute $c_i \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}, a_i)$.
  - Output $c = c_1, \ldots c_n$.
- $\mathsf{ListDec}(\mathsf{sk}, c) \to T$: On input secret key $\mathsf{sk} = \mathsf{sk}_{\mathsf{PRC}}$ and codeword c, the (list) decoding algorithm computes the following:
  - Parse $c = c_1, \ldots c_n$.
  - For each $i \in [n]$, recover a set $S_i \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{ListDec}(\mathsf{sk}_{\mathsf{PRC}}, c_i)$.
  - Compute list $T' \leftarrow \Pi_{\mathsf{LRC}}.\mathsf{Dec}(S_1, \cdots, S_n)$.
  - For each $x \in T'$:
    - Recompute $a_1, \ldots a_n \leftarrow \Pi_{\mathsf{LRC}}.\mathsf{Enc}(x)$.
    - For each $i \in [n]$, check if $a_i \in S_i$.
    - Let ε' be the total fraction of indices for which this is true. If $\varepsilon' \geq \varepsilon$, add x to T.
  - Output T.

Theorem 6.6. *Let Π_{PRC} be a public-key [resp., secret-key] CPA pseudorandom list-decodable code that is $(p + \varepsilon, \ell_{\mathsf{PRC}})$-robust and let Π_{LRC} be a $(\varepsilon, \ell := \ell_{\mathsf{PRC}}, L)$-list-recoverable (standard) code that satisfy the conditions in Construction 6.5. Then the code $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ is a public-key [resp., secret-key] CPA list-decodable code that is pseudorandom and (p, L)-robust. Moreover, if the Π_{PRC} has $(p + \varepsilon)$-simulatable decoding then Π also has p-simulatable decoding.*

Proof of Theorem 6.6. Let $\lambda \in \mathbb{N}$ be the security parameter for the code Π_{PRC}.

Pseudorandomness: Note that by construction the pseudorandomness of the code follows directly from a simple hybrid argument, since the codeword is simply the concatenation of n pseudorandom codewords computed with independent randomness, and n is polynomial in λ.

List-decoding p-robust: For a security parameter $\lambda \in \mathbb{N}$, recall the p-robustness experiment $\mathsf{ExptAR}(\mathcal{A}, \lambda)$ (see Definition 3.5). Let $x \in \{0,1\}^k$ denote the challenge chosen by the adversary and let c denote the response and recall that $(a_1, \cdots, a_n) = \Pi_{\mathsf{LRC}}.\mathsf{Enc}(x)$ and $c = (c_1, \cdots, c_n)$ where for every $i \in [n]$, $c_i \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}, a_i)$, and let c' denote the final noisy codeword sent by the $\mathcal{A}$.

Recall that we can assume that $\delta(c, c') \leq p$. For every $i \in [n]$, define the adversary $\mathcal{A}_i$ to be the adversary that acts like $\mathcal{A}$ restricted to the i block. That is for $i \in [n]$, $\mathcal{A}_i$ receives as input pk, internally runs $\mathcal{A}(\mathsf{pk})$ to get x and sends $a_i = \mathsf{Enc}_{\mathsf{LRC}}(x)_i$ to the challenger and receives c_i, it internally constructs the codewords for the remaining indices and sends $c = (c_1, \cdots, c_i, \cdots, c_n)$ to $\mathcal{A}$ and receives the noisy codeword $c' = (c'_1, \ldots, c'_n)$, and outputs the i-th block c'_i. Let

$$p_i := \delta(c_i, c'_i) \in [0,1] \qquad \text{so that} \qquad \delta(c, c') \;=\; \frac{1}{n}\sum_{i=1}^{n} p_i \;\leq\; p.$$

Define the set of *bad* indices

$$B := \{\, i \in [n] : p_i > p + \varepsilon \,\}.$$

By Markov's inequality, we have

$$\frac{|B|}{n} \;=\; \Pr_{i \leftarrow [n]}[p_i > p + \varepsilon] \;\leq\; \frac{\mathbb{E}_{i \leftarrow [n]}[p_i]}{p + \varepsilon} \;\leq\; \frac{p}{p + \varepsilon} \;\leq 1 - \varepsilon.$$

In particular, for at least $\varepsilon \cdot n$ indices i, we have $p_i \leq p + \varepsilon$. For each $i \in [n]$, let

$$S_i \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{PRC}}, c'_i)$$

be the list output by the inner decoder. By $(p+\varepsilon)$-robustness of Π_{PRC}, for every $i \notin B$ we have $a_i \in S_i$ except with negligible probability. Therefore, except with negligible probability, the sequence of lists $(S_1, \ldots, S_n)$ contains the correct outer codeword symbols $(a_1, \ldots, a_n)$ in at least ε fraction of positions. Applying the (ε, ℓ, L)-list-recovery guarantee of Π_{LRC}, we conclude that x is contained in the final output list $T \leftarrow \mathsf{ListRec}(S_1, \ldots, S_n)$, as required.

p-Simulatable Decoding: Consider the following decoding simulator:

- $\mathsf{Sim}(\mathsf{pk}, x, r, c)$: On input public key $\mathsf{pk} = \mathsf{pk}_{\mathsf{PRC}}$, message $x \in \{0,1\}^k$, randomness $r = (r_1, \ldots r_n)$, and codeword $c = (c_1, \ldots, c_n)$, the decoding simulator:
  - Compute $(a_1, \ldots, a_n) \leftarrow \Pi_{\mathsf{LRC}}.\mathsf{Enc}(x)$.
  - For each $i \in [n]$, check if $\Pi_{\mathsf{PRC}}.\mathsf{Sim}(\mathsf{pk}_{\mathsf{PRC}}, a_i, r_i, c_i) = 1$. Let ε' be the total fraction of indices for which this is true.
  - If $\varepsilon' \geq \varepsilon$, output 1, otherwise output 0.

For a security parameter $\lambda \in \mathbb{N}$, recall the p-simulatable decoding experiment $\mathsf{ExptDS}^{(b)}$ (per Definition 3.7). First, note that in the real decoding algorithm, the adversary computes set S_i via $\Pi_{\mathsf{PRC}}.\mathsf{ListDec}$. By p-simulatable decoding of Π_{PRC}, checking $a_i \in S_i$ on the challenge codeword c_i is computationally indistinguishable from querying $\Pi_{\mathsf{PRC}}.\mathsf{Sim}(a_i, r_i, c_i)$ when given randomness r_i. Now, since both ListDec and Sim only add messages whose LRC codewords agree on at least ε fraction of indices, by the correctness of Π_{LRC}, ListDec and Sim will both correctly recover x. □

6.3 Proof of Theorem 6.1

To prove Theorem 6.1, we instantiate the domain extension framework of Theorem 6.6 using Reed–Solomon codes as the outer list-recoverable code and the list-decodable pseudorandom codes from Lemma 6.2 as the inner component. This immediately yields the desired list-decodable construction with polynomial list size.

We rely on the following list-recovery result for Reed–Solomon codes due to Sudan [Sud97] and Guruswami–Sudan [GS99].

Theorem 6.7 (Reed–Solomon list recovery, [Sud97, GS99])**.** *Let q be a prime power and let $n \leq q$. There is an explicit code $\Pi_{\mathsf{code}} = (\mathsf{Enc}, \mathsf{ListRec})$ over alphabet $[q]$ (namely a Reed–Solomon code) with $\mathsf{Enc} : \mathbb{F}_q^k \to \mathbb{F}_q^n$ such that for every $\ell \geq 1$ and every $\varepsilon > 0$, the code is (α, ℓ, L)-list-recoverable for*

$$\alpha \;\geq\; \sqrt{\frac{(k-1)\ell}{n}} \;+\; \varepsilon,$$

with list size

$$L \;\leq\; \mathsf{poly}\left(k, \ell, \frac{1}{\varepsilon}\right),$$

and there is a (deterministic) list-recovery algorithm running in $\mathsf{poly}\left(n, q, \ell, \frac{1}{\varepsilon}\right)$ time.

Proof of Theorem 6.1. Let $\lambda \in \mathbb{N}$ be the security parameter, and let $\Pi_{\mathsf{PRC}} = (\mathsf{Enc}, \mathsf{ListDec})$ the zero-bit pseudorandom $(p+\varepsilon)$-robust list-decodable code guaranteed by the assumption. Let L_{PRC} be the polynomial guaranteed by Lemma 6.2, and choose $q(\lambda) \geq (L_{\mathsf{PRC}}(\lambda) \cdot \lambda)^4$ such that $\Pi_{\mathsf{PRC}}.\mathsf{Enc} : [q] \to \{0,1\}^m$ and q is a prime. Let Π_{LRC} be a list-recoverable code guaranteed by Theorem 6.7 such that $\Pi_{\mathsf{LRC}}.\mathsf{Enc} : \{0,1\}^\lambda \to [q(\lambda)]^{q(\lambda)}$. Note that for a constant $\varepsilon' < \varepsilon/2$ by Theorem 6.7 the code is $(\alpha, \ell_{\mathsf{PRC}}, L)$-list-recoverable for

$$\alpha \;:=\; \sqrt{\frac{(\lambda-1)\ell_{\mathsf{PRC}}(\lambda)}{q(\lambda)}} \;+\; \varepsilon/2 \;\leq\; \frac{1}{\lambda} \;+\; \varepsilon/2 \;\leq\; \varepsilon$$

and $L \in \mathsf{poly}(\lambda)$. Thus, Π_{PRC} and Π_{LRC} satisfy the conditions in Theorem 6.6 since $\alpha \leq \varepsilon$, thus for a security parameter $\lambda \in \mathbb{N}$, there exists a code Π from λ bits to $n(\lambda)$ bits that is pseudorandom and p-robust list-decodable with list-size $L(\lambda)$ where $L, n \in$ poly and satisfies simulatable decoding. This concludes the proof. □

7 Public Key Shannon Rate Codes

In this section, we construct public-key pseudorandom codes which achieve the Shannon rate. We first give a public-key CPA secure list-decodable code that is p-robust (for $0 \le p < 1/2$) and pseudorandom.

Theorem 7.1. *Let $p, \varepsilon > 0$ be constants such that $p < 1/2$ and let $\lambda \in \mathbb{N}$ be a security parameter. Suppose there exists a public-key zero-bit CPA code Π_{PRC} that is pseudorandom and $(p+\varepsilon)$-robust. Then there exists a polynomial L and a public-key CPA list-decodable pseudorandom and $(p, L(\lambda))$-robust code that achieves rate $R = 1 - H(p) - \varepsilon$.*

Our second result shows that we can have a CCA secure code that approaches the Shannon bound in the **random oracle model**. Moreover, when $p < 1/4$ the construction achieves unique decoding.

Theorem 7.2. *Let $p, \varepsilon > 0$ be constants such that $p < 1/2$ and let $\lambda \in \mathbb{N}$ be a security parameter. Suppose there exists a public-key zero-bit CPA code that is pseudorandom and $(p+\varepsilon)$-robust. Then in the random oracle model: There exists a polynomial L and a public-key CCA list-decodable pseudorandom and $(p, L(\lambda))$-robust code that achieves rate $R = 1 - H(p) - \varepsilon$. Moreover, for $p < 1/4$ the code is uniquely decodable.*

We prove Theorem 7.1 in Section 7.1. The proof of Theorem 7.9 (unique decoding in the plain model) is given in Section 7.2, and the proof for Theorem 7.2 (CCA security in the random oracle model) is given in Section 7.3.

7.1 Construction for Optimal Rate CPA List-Decodable Codes

In this section we prove Theorem 7.1. We start by stating the construction.

Construction 7.3. Let $\lambda \in \mathbb{N}$ be a security parameter and let $0 < p, \alpha$ be constants such that $p + \alpha < 1/2$, and let $m, n \in \mathsf{poly}(\lambda)$ such that $n > m^2$. We makes use of the following ingredients:

- Let $\Pi_{\mathsf{PRC}} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{ListDec})$ be a pseudorandom list-decodable code which is $(p+\alpha, L_{\mathsf{PRC}})$-robust with message space $\{0,1\}^\lambda$ and codeword $\{0,1\}^m$.
- Let $\Pi_{\mathsf{BSC}} = (\mathsf{Enc}, \mathsf{Dec})$ be a code from $\{0,1\}^k$ to $\{0,1\}^n$ that decodes from permuted errors with $(p+\alpha)$-relative weight (see, Theorem 3.4).
- Let PRG_π be a pseudorandom generator with seed length λ and output $\pi : [n] \to [n]$, and let PRG_M be a pseudorandom generator with seed length λ and output $\{0,1\}^n$. And let PRG denote the pseudorandom generator that on seed $s = (s_\pi, s_M) \in \{0,1\}^{2\lambda}$, $\mathsf{PRG}(s) = ((\mathsf{PRG}_\pi(s_\pi), \mathsf{PRG}_M(s_M))$.

We now describe the new code $\Pi = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$:

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: On input security parameter λ, the setup algorithm simply runs the $(\mathsf{pk}_{\mathsf{PRC}}, \mathsf{sk}_{\mathsf{PRC}}) \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{Setup}(1^\lambda)$ and outputs $(\mathsf{pk} = \mathsf{pk}_{\mathsf{PRC}}, \mathsf{sk} = \mathsf{sk}_{\mathsf{PRC}})$.
- $\mathsf{Enc}(\mathsf{pk}, x) \to c$: On input public key $\mathsf{pk} = \mathsf{pk}_{\mathsf{PRC}}$ and message $x \in \{0,1\}^k$, the encoding algorithms computes the following:
  - Sample PRG seed $s \leftarrow \{0,1\}^{2\lambda}$.
  - Derive $(\pi, M) \leftarrow \mathsf{PRG}(s)$, where $\pi : [n] \to [n]$ and $M \in \{0,1\}^n$.
  - Let $q \coloneqq \lfloor \frac{n}{m} \rfloor$, and sample random subset $T \subseteq [q]$ where $|T| = \alpha/10 \cdot q$.
  - Compute $c_{\mathsf{BSC}} \leftarrow \Pi_{\mathsf{BSC}}.\mathsf{Enc}(x)$.
  - Apply permuting and masking $c' = \pi(M \oplus c_{\mathsf{BSC}})$.
  - Divide $c' = c'_1, \ldots, c'_q$ as q blocks of length m.
  - Compute and output $c = c_1, \ldots c_q$, where for each $i \in [q]$,

$$c_i = \begin{cases} c'_i & i \notin T \\ \Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}, s) & i \in T \end{cases}$$

    using fresh randomness for each PRC codeword.

- $\mathsf{ListDec}(\mathsf{sk}, c) \to L$: On input secret key $\mathsf{sk} = \mathsf{sk}_{\mathsf{PRC}}$ and codeword c, the decoding algorithm computes the following:
  - Parse $c = c_1, \ldots, c_q$.
  - For each $i \in [q]$, recover list $L_i \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{ListDec}(\mathsf{sk}_{\mathsf{PRC}}, c_i)$ and define the set:
$$L = \{ s : |\{ i \in [q] : s \in L_i \}| \geq |T| \cdot \alpha/5 \}.$$
  - For each $s \in L$, derive $(\pi, M) \leftarrow \mathsf{PRG}(s)$ and recover $x \leftarrow \mathsf{BSC.Dec}(\pi^{-1}(c) \oplus M)$. If $\delta(\pi(\mathsf{BSC.Enc}(x) \oplus M), c) \leq p$, append x to output list L_{out}.
  - Output L_{out}.

Lemma 7.4 (Robustness, pseudorandomness and rate for the public key construction)**.** *Let $0 < p, \alpha$ be constants with $p + \alpha < 1/2$. Assume the ingredients in Construction 7.3. Then the code Π defined in Construction 7.3 is a public-key CPA list-decodable (p, L)-robust, pseudorandom code with rate $R = k/n$, where $L = c \cdot L_{\mathsf{PRC}}$ and c is a universal constant that depends on α.*

Proof of Theorem 7.1 The proof of Lemma 7.4 is given below, but first we make use of Lemma 7.4 to prove Theorem 7.1.

Proof of Theorem 7.1. For constants $0 < p, \varepsilon$ such that $p < 1/2$, recall that that we are given a public-key CPA zero-bit code that is pseudorandom and $(p + \varepsilon)$-robust. By Theorem 6.1 there exists a polynomial L_{PRC} and a public-key CPA code Π_{PRC} that is pseudorandom and $(p + \varepsilon', L_{\mathsf{PRC}}(\lambda))$-robust for $\varepsilon' = \varepsilon/2$. Let $0 < \alpha \leq \varepsilon'$ be a sufficiently small constant such that
$$1 - H(p + \alpha) - \alpha \geq 1 - H(p) - \varepsilon.$$
Note that since $\alpha < \varepsilon'$ this code is also $(p + \alpha)$-robust with a polynomial size list. Let Π_{BSC} be the code guaranteed in Theorem 3.4 from k bits to n bits (such that n and k are sufficiently large such that $n > m^2$ with rate $1 - H(p + \alpha) - \alpha$, that recovers from permuted errors of $(p + \alpha)$-relative weight. Let Π be the code defined in Construction 4.2 using ingredient Π_{PRC}, Π_{BSC} and PRG (the existence of the PRG is implied by Π_{PRC}). By Lemma 4.3 it follows that the Π is a CPA pseudorandom, $(p, L \coloneqq c \cdot L_{\mathsf{PRC}})$-robust list-decodable code, with rate $1 - H(p) - \varepsilon$. This satisfies the proof. □

7.1.1 Proof of Lemma 7.4

This section is dedicated to proving Lemma 7.4. We make use of the following notation. In Construction 7.3, the public key is $\mathsf{pk} = \mathsf{pk}_{\mathsf{PRC}}$, and $\mathsf{sk} = (\mathsf{sk}_{\mathsf{PRC}}, s \coloneqq (s_\pi, s_M))$ where s_π and s_M are independent and $T \subset [q]$ denotes the subset chosen in the encoding. Recall also that $q \coloneqq \frac{n}{m}$, $T \subseteq [q]$ and that $|T| = \alpha/10 \cdot q$.

Rate and list size. By construction the code has rate $\frac{k}{n}$. Moreover, by construction,
$$|L_{\mathsf{out}}| = |L| \leq \frac{|L_i| \cdot q}{|T| \cdot \alpha/5} \leq \frac{|L_i| \cdot q}{q \cdot \alpha/10 \cdot \alpha/5} \leq \frac{|L_i| \cdot 50}{\alpha^2}$$
it follows that $|L_{\mathsf{out}}| \leq c|L_i| < c \cdot L_{\mathsf{PRC}}$, where $c = 50/\alpha^2$. This concludes the proof for the list size since α is a constant that depends only on ε.

The code is pseudorandom. In the same vein as our secret key code, we will prove a mildly stronger definition of pseudorandomness specific to our construction, which can be directly used later when proving robustness.

For a security parameter $\lambda \in \mathbb{N}$ and a efficient adversary $\mathcal{A}$, we define the following experimenter:

$\mathsf{StrongExptPR}^{(b)}_{\mathsf{pk}}(\mathcal{A}, \lambda)$

1. On input security parameter 1^λ, the challenger samples $\mathsf{pk} = \mathsf{pk}_{\mathsf{PRC}}, \mathsf{sk}_{\mathsf{PRC}} \leftarrow \mathsf{Setup}(1^\lambda)$. The challenger sends pk to the adversary.

2. The adversary submits a challenge $x \in \{0,1\}^k$. Recall the first steps of $\Pi.\mathsf{Enc}$ is sampling $s = (s_\pi, s_M)$ used to derive a pseudorandom permutation π, mask $M \in \{0,1\}^n$, and sampling a random $T \subset [q]$. The challenger always samples these.
   (a) If $b = 0$, the challenger uses this randomness to proceed running the remainder of $c \leftarrow \Pi.\mathsf{Enc}(\mathsf{pk}, x)$.
   (b) If $b = 1$, the challenger ignores this randomness and samples $c \leftarrow \{0,1\}^n$.
   (c) In either case, the challenger sends c with s_π and T to $\mathcal{A}$.
3. The adversary $\mathcal{A}$ submits a guess bit b', which is the output of the experiment.

We prove that

Claim 7.5. $\left|\Pr[\mathsf{StrongExptPR}^{(0)}_{\mathsf{pk}}(\mathcal{A},\lambda)] - \Pr[\mathsf{StrongExptPR}^{(1)}_{\mathsf{pk}}(\mathcal{A},\lambda)]\right| \leq \mathsf{negl}(\lambda)$

Analogous to the secret key case, this is clearly a strict strengthening of the pseudorandomness experiment, as any adversary in the above experiment can be reduced to a (standard) pseudorandomness adversary by simply ignoring the additional permutation description π.

Proof of Claim 7.5. Consider the following simple hybrid

H_0: Same as $\mathsf{StrongExptPR}^{(0)}_{\mathsf{pk}}(\mathcal{A},\lambda)$, except for each of the blocks c_i for $i \in [T]$ of the challenge query, we replace the encoding $\Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}, s)$ with a truly uniform string $u \leftarrow \{0,1\}^m$. By pseudorandomness of Π_{PRC}

$$\left|\Pr[\mathsf{StrongExptPR}^{(0)}_{\mathsf{pk}}(\mathcal{A},\lambda)] - \Pr[H_0]\right| \leq \mathsf{negl}(\lambda).$$

H_1: Same as H_0, except replace the pseudorandom string $M = \mathsf{PRG}_\pi(s_\pi)$ sampled by the challenger by a truly random $M \leftarrow \{0,1\}^n$. Since the challenge is masked by a uniformly random M this means H_1 is identical to $\mathsf{StrongExptPR}^{(1)}_{\mathsf{pk}}(\mathcal{A},\lambda)$. By the security of the PRG,

$$\left|\Pr[H_0] - \Pr[H_1]\right| \leq \mathsf{negl}(\lambda)$$

The above implies that

$$\left|\Pr[\mathsf{StrongExptPR}^{(0)}_{\mathsf{pk}}(\mathcal{A},\lambda)] - \Pr[\mathsf{StrongExptPR}^{(1)}_{\mathsf{pk}}(\mathcal{A},\lambda)]\right| = \mathsf{negl}(\lambda).$$

which concludes the proof. □

The code list-decodes from p fraction of errors. For fixed security parameter $\lambda \in \mathbb{N}$ and efficient adversary $\mathcal{A}$ let $\mathsf{ExptAR} \coloneqq \mathsf{ExptAR}(\mathcal{A},\lambda)$ denote the robustness experiment as defined in Definition 3.5. That is ExptAR,

ExptAR :

1. On input security parameter 1^λ, the challenger samples $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$. The challenger sends pk to the adversary.
2. The adversary submits a challenge $x \in \{0,1\}^k$. The challenger samples $c \leftarrow \mathsf{Enc}(\mathsf{pk}, x)$ using independent randomness.
3. The adversary $\mathcal{A}$ submits a string c'. The output of the experiment is 1 iff $\delta(c', c) \leq p$ and $\mathsf{ListDec}(\mathsf{sk}, c') \notin x$ (and 0 otherwise).

Alternative decoding predicate. For an error pattern e and a set T, we define a modified error pattern $e^T \in \{0,1\}^{qm}$ blockwise by

$$e_i^T = \begin{cases} \mathbf{1}^m & \text{if } i \in T, \\ e_i & \text{if } i \notin T. \end{cases}$$

For a message x, seed s_π, set T and an error pattern e, define $f(x, s_\pi, T, e) = 1$ if and only if the following conditions hold:

1. $|\{i \in T : \delta(e_i) \leq p + \alpha\}| \geq |T| \cdot \alpha/5$
2. For $\pi = \mathsf{PRG}(s_\pi)$, it holds that $\Pi_{\mathsf{BSC}}.\mathsf{Dec}\left(\Pi_{\mathsf{BSC}}.\mathsf{Enc}(x) \oplus \pi^{-1}(e^T)\right) = x$.

Otherwise, define $f(x, s_\pi, T, e) = 0$.

Define the alternative decoding experiment Exp^f as follows:

Exp^f:

1. On input security parameter 1^λ, the challenger samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$. The challenger sends pk to the adversary.
2. The adversary submits a challenge $x \in \{0,1\}^k$. Recall the first steps of $\Pi.\mathsf{Enc}$ is sampling $s = (s_\pi, s_M)$ used to derive a pseudorandom permutation π, mask $M \in \{0,1\}^n$, and sampling a random $T \subset [q]$. The challenger computes $c \leftarrow \mathsf{Enc}(\mathsf{pk}, x)$ using this randomness.
3. The adversary $\mathcal{A}$ submits a string c'. The output of the experiment is 1 iff $\delta(c', c) \leq p$ and

$$f(x, s_\pi, T, e \coloneqq c \oplus c') \neq 1.$$

Finally, consider the experiment $\mathsf{Exp}^{f,ps}$, define similarly to Exp^f, but where the challenge c is uniformly sampled. That is,

$\mathsf{Exp}^{f,ps}$:

1. On input security parameter 1^λ, the challenger samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$. The challenger sends pk to the adversary.
2. The adversary submits a challenge $x \in \{0,1\}^k$. The challenger samples s_π used to derive a pseudorandom permutation π a random $T \subset [q]$. The challenger samples a uniform independent string $c \leftarrow \{0,1\}^n$ and sends it to the adversary.
3. The adversary $\mathcal{A}$ submits a string c'. The output of the experiment is 1 iff $\delta(c', c) \leq p$ and

$$f(x, s_\pi, T, e \coloneqq c \oplus c') \neq 1.$$

The proof for robustness is immediate from the following claims:

Claim 7.6 (Public key variant)**.** $\Pr[\mathsf{Exp}^f] \geq \Pr[\mathsf{ExptAR}] - \mathsf{negl}(\lambda)$.

Claim 7.7 (Public key variant)**.** $\left|\Pr[\mathsf{Exp}^f] - \Pr[\mathsf{Exp}^{f,ps}]\right| \leq \mathsf{negl}(\lambda)$.

Claim 7.8 (Public key variant)**.** $\Pr[\mathsf{Exp}^{f,ps}] \leq \mathsf{negl}(\lambda)$.

Putting together Claims 7.6 to 7.8 we conclude that $\Pr[\mathsf{ExptAR}] = \mathsf{negl}$ as desired. Thus it suffices to prove Claims 7.6 to 7.8 which we do below.

Proof of Claim 7.6. The experiments ExptAR and Exp^f are identical except for the final predicate. In both experiments, the challenger samples x, computes $c \leftarrow \mathsf{Enc}(\mathsf{pk}, x)$, the adversary outputs c', and we define the induced error pattern

$$e \coloneqq c \oplus c'.$$

Conditioned on the event $\delta(c', c) \le p$, we have $\mathsf{ExptAR} = 1$ iff $\mathsf{Dec}(\mathsf{sk}, c') \neq x$, whereas $\mathsf{Exp}^{f} = 1$ iff $f(x, s_\pi, T, e) \neq 1$. Hence it suffices to show that

$$f(x, s_\pi, T, e) = 1 \quad \Longrightarrow \quad \mathsf{Dec}(\mathsf{sk}, c') = x$$

except with negligible probability.

Assume $f(x, s_\pi, T, e) = 1$. By definition of f,

1. $|\{i \in T : \delta(e_i) \le p + \alpha\}| \ge |T| \cdot \alpha/5$
2. For $\pi = \mathsf{PRG}(s_\pi)$, it holds that $\Pi_{\mathsf{BSC}}.\mathsf{Dec}\left(\Pi_{\mathsf{BSC}}.\mathsf{Enc}(x) \oplus \pi^{-1}(e^T)\right) = x$.

Now consider $\mathsf{Dec}(\mathsf{sk}, c')$. The decoder does the following: for every $i \in [q]$, recover list $L_i \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{ListDec}(\mathsf{sk}_{\mathsf{PRC}}, c_i)$ and computes:

$$L \;=\; \{\, x : |\{\, i \in [q] : x \in L_i \,\}| \ge |T| \cdot \alpha/5\},$$

and for each $s \in L$, derive $(\pi, M) \leftarrow \mathsf{PRG}(s)$ and recover $x \leftarrow \Pi_{\mathsf{BSC}}.\mathsf{Dec}(\pi^{-1}(c) \oplus M)$ and outputs L_{out} the set of x recovered in the previous step.

By our definition of f, since for at least $|T| \cdot \alpha/5$ of the indices in $i \in T$, $\delta(e_i) \le p + \alpha$, by $(p + \alpha)$-robustness of Π_{PRC} it follows that s is successfully recovered with all but negligible probability. Thus, when unmasking and unpermuting using $(\pi, M) = \mathsf{PRG}(s)$, it holds that

$$\Pi_{\mathsf{BSC}}.\mathsf{Dec}(\pi^{-1}(c) \oplus M) = \Pi_{\mathsf{BSC}}.\mathsf{Dec}(\Pi_{\mathsf{BSC}}.\mathsf{Enc}(x) \oplus \pi^{-1}(e')),$$

where $e' \in \{0,1\}^n$ is an error pattern for which for every $j \in [n]$, $e'_j \le e^T_j$, thus by the "monotone decoding" property of the Π_{BSC} (see Theorem 3.4),

$$\Pi_{\mathsf{BSC}}.\mathsf{Dec}(\Pi_{\mathsf{BSC}}.\mathsf{Enc}(x) \oplus \pi^{-1}(e^T)) = x \Rightarrow \Pi_{\mathsf{BSC}}.\mathsf{Dec}(\Pi_{\mathsf{BSC}}.\mathsf{Enc}(x) \oplus \pi^{-1}(e')) = x.$$

Where the above implication follows by the second item in the definition of f. This implies that $x \in \mathsf{ListDec}(c)$ with all but negligible probability, concluding the proof. □

Proof of Claim 7.7. The claim follows directly from the strong pseudorandomness property established in Claim 7.5. In particular, the pseudorandomness guarantee holds even when the adversary is given the permutations s_π (and π which can be efficiently computed from s_π) and T.

Observe that in both experiments Exp^{f} and $\mathsf{Exp}^{f,ps}$ the winning condition is identical, namely $f(x, s_\pi, T, e) \neq 1$, where $e = c \oplus c'$. Moreover, this predicate depends only on (x, s_π, T, e) (and does not require the knowledge of the PRC secret key, or the seed of the pseudorandomness generator s_M).

Therefore, by the strong pseudorandomness property, the joint distributions of (x, s_π, T, e) in the two experiments are computationally indistinguishable, and hence

$$\left|\Pr[\mathsf{Exp}^{f}] - \Pr[\mathsf{Exp}^{f,ps}]\right| \le \mathsf{negl}(\lambda),$$

which proves the claim. □

Proof Claim 7.8. Fix any PPT adversary $\mathcal{A}$ and consider the experiment $\mathsf{Exp}^{f,ps}$. Let $c \in \{0,1\}^n$ be the (uniform) challenge string and let c' be the adversary's output. Define the induced error vector

$$e \coloneqq c \oplus c'.$$

Notice that in the experiment $\mathsf{Exp}^{f,ps}$ the permutations seed π_s and set T are random and independent from e and c. Let $\pi = \mathsf{PRG}(s_\pi)$ and note that by a simple hybrid argument for all but negligible probability we can assume that π is a fully random and independent permutation. Recall that, by definition $f(x, s_\pi, T, e) = 1$ if

1. $|\{i \in T : \delta(e_i) \le p + \alpha\}| \ge |T| \cdot \alpha/5$
2. For $\pi = \mathsf{PRG}(s_\pi)$, it holds that $\Pi_{\mathsf{BSC}}.\mathsf{Dec}\left(\Pi_{\mathsf{BSC}}.\mathsf{Enc}(x) \oplus \pi^{-1}(e^T)\right) = x$.

Note that for any fixed error pattern $e \in \{0,1\}^n$, such that $\delta(e) \le p$, by a simple hybrid argument, by the number of blocks $i \in [q]$ for which $\delta(e_i) < p + \alpha$ is at least $\frac{\alpha}{p+\alpha} \ge \alpha$ (since $p < 1/2$). Since T is a random and independent set by a simple Chernoff bound,

$$\Pr\left[\left|\{i \in T : \delta(e_i) \le p + \alpha\}\right| \ge |T| \cdot \alpha/5\right] \le 1 - 2^{-\Omega(\alpha|T|)}. \tag{7.1}$$

Since $\delta(e) \le p$ and $|T| = \alpha/10$, it follows by definition that $\delta(e^T) < p + \alpha$. Thus by the property of the code $(\mathsf{Enc}_{\mathsf{BSC}}, \mathsf{Dec}_{\mathsf{BSC}})$ (see Theorem 3.4), and since π is a random independent permutation it follows that

$$\Pr[\Pi_{\mathsf{BSC}}.\mathsf{Dec}\left(\Pi_{\mathsf{BSC}}.\mathsf{Enc}(x) \oplus \pi^{-1}(e^T)\right) = x] \ge 1 - 2^{-\Omega(n^{0.1})}. \tag{7.2}$$

Putting Eqs. (7.1) and (7.2) together it follows that $\Pr[\mathsf{Exp}^{f,ps}] \le \mathsf{negl}(\lambda)$ (since by our choice of parameters λ is smaller than m and n) this which proves the claim. □

7.2 Optimal Rate CPA Uniquely Decodable Codes in the Plain Model

To achieve unique decoding in the plain model while preserving rate and pseudorandomness, we extend the approach of [SW25a, LSW25]. At a high level, this is done by taking a list-decodable code with good rate and encoding the message as $(x, h(x))$, where $h(x)$ is a short hash of x. To decode, the list-decoding algorithm produces a list $\{(x_i, y_i)\}_i$, which the decoder prunes to output the unique x_i satisfying $y_i = h(x_i)$. By constructing hash functions h satisfying appropriate forms of *correlation intractability*, these prior works argue that it is computationally hard for an adversary to find distinct $x \ne x'$ for which $(x, h(x))$ and $(x', h(x'))$ simultaneously appear in the same decoding list.

However, the prior works [SW25a, LSW25] operate in the setting where the underlying list-decodable code is a traditional deterministic linear code, whereas our Shannon-rate list-decodable code Construction 7.3 is both randomized (a necessity for pseudorandomness) and non-linear. Nevertheless, we show that Construction 7.3, when Π_{BSC} is instantiated with an appropriate linear code, can be made uniquely decodable via an adaptation of this correlation-intractability approach. Compared to [LSW25], this requires strengthening the framework in two ways: (1) the correlation-intractable hash must be *subexponentially secure* (which follows from subexponential variants of the same assumptions used in [LSW25]), and (2) the linear code Π_{BSC} must satisfy a stronger notion of *pairwise column-robustness*, which we show is achievable unconditionally at Shannon rate.

The main result of this section is the following theorem.

Theorem 7.9 (Optimal Rate CPA Uniquely Decodable Codes)**.** *Let $p, \varepsilon > 0$ be constants with $p < 1/4$, and let $\lambda \in \mathbb{N}$ be a security parameter. Suppose there exists a public-key zero-bit CPA code that is pseudorandom and $(p + \varepsilon)$-robust and a subexponentially secure linearly shifted 2-input correlation intractable hash for $s_0 \in \mathit{poly}(\lambda)$-sparse output relations. Then there exists a public-key CPA uniquely decodable pseudorandom code that is p-robust and achieves rate $R = 1 - H(p) - \varepsilon$.*

By instantiating the correlation-intractable hash function with the constructions of Theorem 3.18 from any of {DDH, LWE, QR, DCR}, as well as the public-key PRC of Theorem 3.11 from LPN, this gives us the following concrete instantiation of the above theorem.

Corollary 7.10. *Let $p, \varepsilon > 0$ be constants with $p < 1/4$, and let $\lambda \in \mathbb{N}$ be a security parameter. Assuming constant noise* LPN *with $2^{\sqrt{n}}$ security and any of* {DDH, LWE, QR, DCR} *with subexponential security, there exists a public-key CPA uniquely decodable pseudorandom code that is p-robust and achieves rate $R = 1 - H(p) - \varepsilon$.*

We prove Theorem 7.9 at the end of this section by composing the building blocks developed below and in Section 3.3.

7.2.1 Decomposable Binary Linear Codes with Pairwise Column-Robustness

We recall the structural notion of decomposable binary linear codes that will serve as the inner code Π_{BSC} in our construction. The high level idea is that the generator matrix decomposes into a "message part" $\mathbf{A}$ and a "hash part" $\mathbf{B}$. We will require a robustness property on $\mathbf{B}$ that is tailored to the pairwise structure arising from two independent permutations.

Definition 7.11 (Decomposable Binary Linear Code)**.** A **decomposable binary linear code** with message length $k \in \mathbb{N}$ and nested message length $v \in \mathbb{N}$ is specified by a deterministic, efficiently computable generating matrix

$$\mathbf{G} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix} \in \mathbb{F}_2^{(k+v)\times n},$$

where $\mathbf{A} \in \mathbb{F}_2^{k\times n}$, $\mathbf{B} \in \mathbb{F}_2^{v\times n}$, and $\mathbf{G}$ has full rank. We write $\mathsf{Enc}_{\mathbf{G}}((m, y) \in \{0,1\}^{k+v}) = (m, y) \cdot \mathbf{G}$ for the induced encoding map, and define the **rate** of the code to be $R = (k + v)/n$.

Definition 7.12 (Pairwise Column-Robustness)**.** Let $\mathbf{G} = \left[\begin{smallmatrix} \mathbf{A} \\ \mathbf{B} \end{smallmatrix}\right]$ be a decomposable binary linear code (Definition 7.11), and let $\alpha > 0$ be a constant. Write $w := \lceil v^\alpha \rceil$. We say $\mathbf{B}$ is (q, α)**-pairwise well-formed** (for a polynomial $q : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$) if for every pair of permutations $\pi_1, \pi_2 \in \mathsf{Perm}(n)$, sampling a uniformly random subset $S \subseteq [n]$ with $|S| = w$ and defining $S^* := \pi_1^{-1}(S) \cup \pi_2^{-1}(S)$,

$$\Pr_S\left[\mathsf{rank}\left([\mathbf{B}]_{S^*}\right) = |S^*|\right] \;\geq\; \frac{1}{q(k, v)}.$$

We say the decomposable code $\mathbf{G}$ is α**-pairwise column-robust** if there exists a polynomial q such that $\mathbf{B}$ is (q, α)-pairwise well-formed.

Remark 7.13. Note that $|S^*| \leq 2w$ always holds, and $|S^*| = 2w$ whenever $\pi_1^{-1}(S) \cap \pi_2^{-1}(S) = \emptyset$. The definition requires full rank of $[\mathbf{B}]_{S^*}$ regardless of the overlap structure of the two permuted copies. When $\pi_1 = \pi_2$, this reduces to requiring rank w for a single random subset, recovering the standard (non-pairwise) column-robustness notion of [LSW25].

Theorem 7.14 (Shannon-Rate Decomposable Code with Pairwise Column-Robustness)**.** *For every constant $0 \leq p < 1/2$ and every sufficiently small $\varepsilon > 0$, there exists a constant $\alpha > 0$ and an explicit family of decomposable binary linear codes $\{\mathbf{G}_{k,v}\}$ (Definition 7.11), indexed by message length k and nested message length v, satisfying:*

1. ***Rate:*** $(k + v)/n \geq 1 - H(p) - \varepsilon$.
2. α***-Pairwise Column-Robustness:*** *For all sufficiently large k and v in range $\log^5(k) < v < k^{0.49}$, the code $\mathbf{G}_{k,v}$ is α-pairwise column-robust (Definition 7.12).*
3. ***Efficient Decodability from Permuted Errors:*** *The code $(\mathsf{Enc}_{\mathbf{G}_{k,v}}, \mathsf{Dec}_{\mathbf{G}_{k,v}})$ decodes from permuted errors of relative weight p (in the sense of Theorem 3.4).*

The proof of Theorem 7.14 is given in Section A.

7.2.2 Correlation Intractability Against Permuted and Masked Linear Codes

In this subsection we show that linearly shifted 2-MICIH implies correlation intractability against the relation induced by two codewords of a decomposable binary linear code that are close after independent permutations and masks are applied. This is the key technical lemma needed to argue unique decodability of our construction.

Definition 7.15 (Permuted-Masked Linear Code Relation)**.** Let $\mathbf{G} = \left[\begin{smallmatrix} \mathbf{A} \\ \mathbf{B} \end{smallmatrix}\right] \in \mathbb{F}_2^{(k+v)\times n}$ be the generator matrix of a decomposable binary linear code, let $M_1, M_2 \in \{0,1\}^n$, $\pi_1, \pi_2 \in \mathsf{Perm}(n)$, and $p \in [0, \frac{1}{2})$. Define the relation $\mathcal{R}_{\mathbf{G},\pi_1,\pi_2,M_1,M_2,p}$ on pairs $((x_1, y_1), (x_2, y_2)) \in (\{0,1\}^k \times \{0,1\}^v)^2$ by

$$\mathcal{R}_{\mathbf{G},\pi_1,\pi_2,M_1,M_2,p}\big((x_1, y_1), (x_2, y_2)\big) = 1 \;\Leftrightarrow\; \delta\big(\pi_1([x_1, y_1] \cdot \mathbf{G} \oplus M_1),\; \pi_2([x_2, y_2] \cdot \mathbf{G} \oplus M_2)\big) \leq p \;\wedge\; (x_1, y_1) \neq (x_2, y_2).$$

Lemma 7.16 (CI Against Permuted-Masked Linear Codes)**.** *Let $\mathbf{G} = \left[\begin{smallmatrix} \mathbf{A} \\ \mathbf{B} \end{smallmatrix}\right]$ be an α-pairwise column-robust decomposable binary linear code (Definitions 7.11 and 7.12) with nested message length v, and let Π_{Hash} be a linearly shifted 2-MICIH for $s_0(\lambda_{\mathsf{Hash}})$-sparse output relations (Definition 3.17 and Theorem 3.18). Suppose v is chosen large enough that $\Omega(v^\alpha) \geq s_0(\lambda_{\mathsf{Hash}})$ (i.e., the sparsity of the relation induced by the code exceeds the sparsity threshold of the hash).*

Then for any fixed $M_1, M_2 \in \{0,1\}^n$, $\pi_1, \pi_2 \in \mathsf{Perm}(n)$, and constant $p < 1/2$, Π_{Hash} is correlation intractable against $\mathcal{R}_{\mathbf{G},\pi_1,\pi_2,M_1,M_2,p}$.

More precisely, for all polynomial-time adversaries $\mathcal{A}$,

$$\Pr\Big[K \leftarrow \mathsf{Gen}(1^{\lambda_{\mathsf{Hash}}}, 1^k, 1^v);\ (x_1, y_1, x_2, y_2) \leftarrow \mathcal{A}(K) :\ \mathcal{R}_{\mathbf{G},\pi_1,\pi_2,M_1,M_2,p}\big((x_1, y_1), (x_2, y_2)\big) = 1\Big] = \mathsf{negl}(\lambda_{\mathsf{Hash}}).$$

Note that this lemma requires only standard (polynomial) CI security, since $\mathbf{G}$, π_1, π_2, M_1, M_2 are all fixed and the resulting relation is a single efficiently checkable linearly shifted output relation. The condition $\Omega(v^\alpha) \geq s_0(\lambda_{\mathsf{Hash}})$ ensures the relation is sparse enough for the hash to handle. Subexponential security will only be needed later in the compilation (Section 7.2.3), where the reduction must guess the PRG seeds.

Proof. We proceed via a sequence of hybrid experiments. Throughout, fix $\mathbf{G} = \left[\begin{smallmatrix}\mathbf{A}\\ \mathbf{B}\end{smallmatrix}\right]$, M_1, M_2, π_1, π_2, and $p < 1/2$.

Experiment Real**.** This is the real correlation-intractability experiment: the challenger samples $K \leftarrow \mathsf{Gen}(1^\lambda, 1^k, 1^v)$, sends K to $\mathcal{A}$, receives (x_1, y_1, x_2, y_2), computes $h_i = \mathsf{Hash}(K, (x_i, y_i))$ for $i \in \{1, 2\}$, and outputs 1 iff

$$\mathcal{R}_{\mathbf{G},\pi_1,\pi_2,M_1,M_2,p}((x_1, y_1), (x_2, y_2)) = 1.$$

Experiment H_1**.** Identical to Real, except the challenger additionally samples a uniformly random subset $S \subseteq [n]$ of size $w := \lceil v^\alpha \rceil$, and defines $S^* := \pi_1^{-1}(S) \cup \pi_2^{-1}(S) \subseteq [n]$. The challenger aborts (outputting 0) if $[\mathbf{B}]_{S^*}$ does not have full column rank (i.e., $\mathsf{rank}([\mathbf{B}]_{S^*}) < |S^*|$).

Claim 7.17. $\Pr[\mathsf{Real} = 1] \leq q(k, v) \cdot \Pr[H_1 = 1]$*, where q is the polynomial from the (q, α)-pairwise well-formedness of* $\mathbf{B}$ *(Definition 7.12).*

Proof. Since $\mathbf{G}$ is α-pairwise column-robust, $\mathbf{B}$ is (q, α)-pairwise well-formed for some polynomial q. For the fixed permutations π_1, π_2 and a uniformly random S of size w, the columns $[\mathbf{B}]_{S^*}$ have rank $|S^*|$ with probability at least $1/q(k, v)$. Hence the abort event in H_1 occurs with probability at most $1 - 1/q(k, v)$, and conditioned on not aborting, H_1 is identical to Real. The claim follows. □

Experiment H_2**.** Identical to H_1, except the challenger replaces the global Hamming distance check $\delta(\pi_1(c_1 \oplus M_1), \pi_2(c_2 \oplus M_2)) \leq p$ (where $c_i = [x_i, y_i] \cdot \mathbf{G}$) with the localized check: the fraction of positions in S where $(\pi_1(c_1 \oplus M_1))_j \neq (\pi_2(c_2 \oplus M_2))_j$ is at most $p + \varepsilon$ for a small constant $\varepsilon > 0$ to be chosen.

Claim 7.18. $\Pr[H_2 = 1] \leq \Pr[H_1 = 1] + \mathsf{negl}(\lambda)$.

Proof. Since S is a uniformly random subset of $[n]$ of size w, chosen independently of the adversary's output (x_1, y_1, x_2, y_2). Let $d = \delta(\pi_1(c_1 \oplus M_1), \pi_2(c_2 \oplus M_2))$ be the global relative Hamming distance. By a Chernoff bound, for any fixed strings, the fraction of disagreements on a random subset S of size w concentrates around d: for any $\varepsilon > 0$,

$$\Pr_S\left[\left|\frac{|\{j \in S : (\pi_1(c_1 \oplus M_1))_j \neq (\pi_2(c_2 \oplus M_2))_j\}|}{w} - d\right| > \varepsilon\right] \leq 2\exp(-2\varepsilon^2 w).$$

Since $w = \lceil v^\alpha \rceil$ grows with λ (as $v = v(\lambda)$), this probability is negligible. Since this is exactly the event H_2 returns 1 while H_1 does not, this gives us our claim. □

Experiment H_3**.** Identical to H_2, except the challenger rewrites the localized distance condition as the following shifted output relation:

$$f(x) = x \cdot [\mathbf{A}]_{S^*} \cdot [\mathbf{B}]_{S^*}^{-1} \quad , \quad R_{\mathsf{out}} = \Big\{(z_1, z_2) \in (\{0,1\}^v)^2 : \delta\big(z_1 \cdot [\mathbf{B}]_{S_1^*} \oplus [M_1]_{S_1^*},\ z_2 \cdot [\mathbf{B}]_{S_2^*} \oplus [M_2]_{S_2^*}\big) \leq p + \varepsilon\Big\}.$$

Where $S_i^* = \pi_i^{-1}(S) \subseteq S^*$, $[\mathbf{B}]_{S^*}^{-1}$ is a left inverse of $\mathbf{B}_{S^*}$ (which exists as $\mathbf{B}_{S^*}$ has full column rank).

Claim 7.19. $\Pr[H_2 = 1] = \Pr[H_3 = 1]$.

Proof. The two experiments differ only syntactically and check the same condition. To see this, we can substitute $z_i = y_i + f(x_i)$, which allows us to rewrite:

$$\begin{aligned} z_i \cdot \mathbf{B}_{S_i^*} \oplus [M_i]_{S_i^*} &= \left(y_i + x_i \cdot [\mathbf{A}]_{S^*} \cdot [\mathbf{B}]_{S^*}^{-1}\right) \cdot [\mathbf{B}]_{S_i^*} \oplus [M_i]_{S_i^*} \\ &= y_i \cdot [\mathbf{B}]_{S_i^*} + x_i \cdot [\mathbf{A}]_{S^*} \cdot [\mathbf{B}]_{S^*}^{-1} \cdot [\mathbf{B}]_{S_i^*} \oplus [M_i]_{S_i^*} \\ &= y_i \cdot [\mathbf{B}]_{S_i^*} + x_i \cdot [\mathbf{A}]_{S_i^*} \oplus \pi_i([M_i])_S \\ &= \pi_i(y_i \cdot [\mathbf{B}]_{S_i^*} + x_i \cdot [\mathbf{A}]_{S_i^*})_S \oplus \pi_i([M_i])_S \\ &= \pi_i([x_i, y_i] \cdot \mathbf{G} \oplus M_i)_S \\ &= \pi_i(c_i \oplus M_i)_S \end{aligned}$$

which, when in turn substituted into R_{out} gives

$$\delta(\pi_1(c_2 \oplus M_2)_S, \pi_2(c_2 \oplus M_2)_S) \le p + \varepsilon,$$

the exact condition checked in H_2. □

Claim 7.20. $\Pr[H_3 = 1] = \mathsf{negl}(\lambda_{\mathsf{Hash}})$.

Proof. Since an adversary on H_3 is directly an adversary for correlation intractability of a shifted output relation on Π_{Hash}, it suffices to show that the relation R_{out} is sufficiently sparse. More formally, we show that for $p < 1/2$ and sufficiently small $\varepsilon > 0$, the relation R_{out} is s-sparse for $s = \Omega(w) = \Omega(v^\alpha)$.

Fix $z_1 \in \{0,1\}^v$. The number of $z_2 \in \{0,1\}^v$ such that $(z_1, z_2) \in R_{\text{out}}$ equals the number of z_2 such that $z_2 \cdot [\mathbf{B}]_{S_2^*}$ lies within relative Hamming distance $p + \varepsilon$ of the fixed string $z_1[\mathbf{B}]_{S_1^*} \oplus [M_1]_{S_1^*} \oplus [M_2]_{S_2^*}$. Since $[\mathbf{B}]_{S_2^*}$ has rank w, the map $z_2 \mapsto z_2 \cdot [\mathbf{B}]_{S_2^*}$ is a surjection from $\{0,1\}^v$ onto a w-dimensional subspace of $\{0,1\}^w$. The number of w-bit strings within relative Hamming distance $p + \varepsilon$ of any fixed string is at most $2^{H(p+\varepsilon)\cdot w + o(w)}$ by the volume bound. Since $p < 1/2$ implies $H(p + \varepsilon) < 1$ for small enough ε, the fraction of valid z_2 is at most $2^{(H(p+\varepsilon)-1)w} = 2^{-\Omega(w)}$. Summing over z_1 gives $|R_{\text{out}}| \le 2^v \cdot 2^{v-\Omega(w)} = 2^{2v-\Omega(v^\alpha)}$, so $|R_{\text{out}}|/2^{2v} \le 2^{-\Omega(v^\alpha)}$, i.e., R_{out} is $\Omega(v^\alpha)$-sparse.

Since $v = v(\lambda_{\mathsf{Hash}})$ is chosen so that $\Omega(v^\alpha) \ge s_0(\lambda_{\mathsf{Hash}})$, the relation R_{out} is $s_0(\lambda_{\mathsf{Hash}})$-sparse. The relation R_{out} is efficiently checkable (given $\mathbf{G}$, π_1, π_2, M_1, M_2, S, which are all fixed), and the shift functions f_i are $\mathbb{F}_2$-linear. Therefore, the relation $\mathcal{R}$ defined by $(z_1, z_2) \in R_{\text{out}}$ with $z_i = y_i \oplus f_i(x_i)$ is a linearly shifted $s_0(\lambda_{\mathsf{Hash}})$-sparse output relation. By the CI security of Π_{Hash} (Theorem 3.18), the probability that $\mathcal{A}$ wins in H_3 is $\mathsf{negl}(\lambda_{\mathsf{Hash}})$. □

Combining Claims 7.17, 7.18, 7.19, and the 2-MICIH security of Π_{Hash} against the linearly shifted $s_0(\lambda_{\mathsf{Hash}})$-sparse relation in H_3:

$$\Pr[\mathsf{Real} = 1] \le q(k, v) \cdot \Pr[H_3 = 1] \le q(k, v) \cdot \mathsf{negl}(\lambda_{\mathsf{Hash}}) = \mathsf{negl}(\lambda_{\mathsf{Hash}}),$$

where the last step uses that $q(k, v)$ is a fixed polynomial. This completes the proof. □

7.2.3 Construction and Proof of Theorem 7.9

We now present the full construction and prove Theorem 7.9. The key subtlety is that the permutations and masks in Construction 7.3 are derived from PRG seeds that are not known to the reduction a priori, so the reduction must guess them, incurring a $2^{O(\lambda)}$ loss. To absorb this loss, we instantiate the correlation-intractable hash with a sufficiently large (but notably still independent of message length) security parameter.

Construction 7.21 (Optimal Rate CPA Uniquely Decodable Code)**.** Let $\lambda \in \mathbb{N}$ be a security parameter and let $p, \varepsilon > 0$ be constants with $p < 1/4$. The construction uses the following ingredients:

- Let $\Pi_{\mathsf{code}} = (\mathsf{Setup}_{\mathsf{code}}, \mathsf{Enc}_{\mathsf{code}}, \mathsf{ListDec}_{\mathsf{code}})$ be the rate-optimal CPA list-decodable pseudorandom code from Construction 7.3, where Π_{BSC} is instantiated with the explicit decomposable binary linear code $\mathbf{G}$ from Theorem 7.14. The code has message space $\{0,1\}^{k+v}$ (encoding both the message $x \in \{0,1\}^k$ and the hash value $y \in \{0,1\}^v$), codeword space $\{0,1\}^n$, and is (p, L)-robust for a polynomial L. Recall from Construction 7.3 that each encoding samples a PRG seed $s \in \{0,1\}^\lambda$.

- Let $\Pi_{\mathsf{Hash}} = (\mathsf{Gen}_{\mathsf{Hash}}, \mathsf{Hash})$ be a linearly shifted 2-MICIH for $s_0(\lambda_{\mathsf{Hash}})$-sparse output relations (Theorem 3.18), with subexponential security $2^{\lambda_{\mathsf{Hash}}^{c'}}$ for some constant $c' > 0$. We instantiate Π_{Hash} with security parameter $\lambda_{\mathsf{Hash}} := \lambda^{\lceil 2/c' \rceil}$, so that the CI security satisfies $2^{\lambda_{\mathsf{Hash}}^{c'}} \geq 2^{\lambda^2}$. We set the hash output length $v := s_0(\lambda_{\mathsf{Hash}})^{\lceil 1/\alpha \rceil}$, which ensures $v^\alpha \geq s_0(\lambda_{\mathsf{Hash}})$ and thus the sparsity condition of Lemma 7.16 is satisfied. Since s_0 is a polynomial and $\lambda_{\mathsf{Hash}} = \mathsf{poly}(\lambda)$, we have $v = \mathsf{poly}(\lambda)$.

The new code $\Pi = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ is defined as follows:

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: Run $(\mathsf{pk}_{\mathsf{code}}, \mathsf{sk}_{\mathsf{code}}) \leftarrow \Pi_{\mathsf{code}}.\mathsf{Setup}(1^\lambda)$ and $\mathsf{hk} \leftarrow \Pi_{\mathsf{Hash}}.\mathsf{Gen}(1^{\lambda_{\mathsf{Hash}}}, 1^k, 1^v)$. Output $\mathsf{pk} = (\mathsf{pk}_{\mathsf{code}}, \mathsf{hk})$ and $\mathsf{sk} = (\mathsf{sk}_{\mathsf{code}}, \mathsf{hk})$.
- $\mathsf{Enc}(\mathsf{pk}, x) \to c$: On input $\mathsf{pk} = (\mathsf{pk}_{\mathsf{code}}, \mathsf{hk})$ and message $x \in \{0,1\}^k$, compute $y \leftarrow \mathsf{Hash}(\mathsf{hk}, x)$ and output $c \leftarrow \Pi_{\mathsf{code}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{code}}, (x, y))$.
- $\mathsf{Dec}(\mathsf{sk}, c) \to x$ or $\perp$: On input $\mathsf{sk} = (\mathsf{sk}_{\mathsf{code}}, \mathsf{hk})$ and codeword c, compute the list $L \leftarrow \Pi_{\mathsf{code}}.\mathsf{ListDec}(\mathsf{sk}_{\mathsf{code}}, c)$. If there exists a unique $(x_i, y_i) \in L$ such that $\mathsf{Hash}(\mathsf{hk}, x_i) = y_i$, output x_i. Otherwise, output $\perp$.

Proof of Theorem 7.9. We verify that Construction 7.21 satisfies rate, pseudorandomness, and p-robustness with unique decoding.

Rate. The rate of Π_{code} is $1 - H(p) - \varepsilon$ with respect to total message length $k + v$. Since $v = s_0(\lambda_{\mathsf{Hash}})^{\lceil 1/\alpha \rceil} = \mathsf{poly}(\lambda)$ and $k \geq \lambda$, the effective rate with respect to k is

$$R = (1 - H(p) - \varepsilon) \cdot \frac{k}{k + v}.$$

Since $v = \mathsf{poly}(\lambda)$ and $k = k(\lambda) \geq \lambda$, we have $k/(k+v) \to 1$ as $\lambda \to \infty$. Thus for any desired $\varepsilon' > 0$, the rate is at least $1 - H(p) - \varepsilon'$ for all sufficiently large λ (by choosing ε small enough relative to ε'). Note that the hash output length v depends only on λ (through λ_{Hash}, s_0, and α, all of which are determined by the constants p, ε, c'), so the rate loss from appending the hash is $o(1)$.

Pseudorandomness. Codewords of Π are codewords of Π_{code} on input $(x, \mathsf{Hash}(\mathsf{hk}, x))$. Since hk is part of the public key and $\mathsf{Hash}(\mathsf{hk}, x)$ is a deterministic function of x and hk, pseudorandomness of Π follows directly from pseudorandomness of Π_{code}: any adversary distinguishing codewords of Π from uniform can be used to distinguish codewords of Π_{code} from uniform by computing $y = \mathsf{Hash}(\mathsf{hk}, x)$ and forwarding (x, y) as the challenge message.

Unique Decoding (p-Robustness). We show that for any efficient adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins the robustness game is negligible. Recall that $\mathcal{A}$ receives pk, submits challenge x, receives $c = \mathsf{Enc}(\mathsf{pk}, x)$, and outputs c'. The adversary wins if $\delta(c', c) \leq p$ and $\mathsf{Dec}(\mathsf{sk}, c') \neq x$.

We observe by the robustness of the underlying Π_{code} that a decoding failure implies the existence of two hash-consistent pairs in the decoding list.

Claim 7.22. *If $\delta(c', c) \leq p$ and $\mathsf{Dec}(\mathsf{sk}, c') \neq x$, then with probability $1 - \mathsf{negl}(\lambda)$ there exist distinct $(x_1, y_1) \neq (x_2, y_2)$ in $L = \Pi_{\mathsf{code}}.\mathsf{ListDec}(\mathsf{sk}_{\mathsf{code}}, c')$ with $\mathsf{Hash}(\mathsf{hk}, x_i) = y_i$ for $i \in \{1, 2\}$.*

Proof. Since $\delta(c', c) \leq p$ and $c = \Pi_{\mathsf{code}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{code}}, (x, \mathsf{Hash}(\mathsf{hk}, x)))$, by (p, L)-robustness of Π_{code} the pair $(x, \mathsf{Hash}(\mathsf{hk}, x))$ is in L with probability $1 - \mathsf{negl}(\lambda)$. Conditioned on this, if $\mathsf{Dec}(\mathsf{sk}, c') \neq x$, then either no pair in L satisfies $\mathsf{Hash}(\mathsf{hk}, x_i) = y_i$ (impossible, since $(x, \mathsf{Hash}(\mathsf{hk}, x)) \in L$), or there are at least two such pairs. Hence there exists $(x_2, y_2) \in L$ with $(x_2, y_2) \neq (x, \mathsf{Hash}(\mathsf{hk}, x))$ and $\mathsf{Hash}(\mathsf{hk}, x_2) = y_2$. □

We can now show the adversary's winning probability is negligible via two hybrids.

Experiment H_0: This is the real robustness experiment. Let $\varepsilon_{\mathcal{A}} := \Pr[\mathcal{A} \text{ wins in } H_0]$.

Experiment H_1: In this experiment, the challenger takes a subexponential loss in advantage to guess the random seeds of Π_{code}.

Recall by Claim 7.22, conditioned on $\mathcal{A}$ winning, the list L contains two distinct pairs $(x_1, y_1), (x_2, y_2)$ with $\mathsf{Hash}(\mathsf{hk}, x_i) = y_i$. Tracing through the list-decoding algorithm of Construction 7.3, each $(x_i, y_i) \in L$ is associated with a PRG seed $s_i \in \{0,1\}^\lambda$ from which the permutation π_i and mask M_i are derived via $(\pi_i, M_i) = \mathsf{PRG}(s_i)$.

In H_1, the challenger guesses the pair of seeds $(s_1, s_2) \in (\{0,1\}^\lambda)^2$ uniformly at random *before* running $\mathcal{A}$, and aborts if the guess is incorrect (i.e., if the seeds associated with the two hash-consistent pairs in L do not match the guess). Since each seed is λ bits and there are two seeds to guess:

$$\Pr[\mathcal{A} \text{ wins in } H_1] \;\geq\; \frac{\varepsilon_{\mathcal{A}}}{2^\lambda}.$$

Experiment H_2: Conditioned on the correct guess of (s_1, s_2), the permutations π_1, π_2 and masks M_1, M_2 are now *fixed* (determined by the guessed seeds). The adversary $\mathcal{A}$ has produced $(x_1, y_1) \neq (x_2, y_2)$ with $\mathsf{Hash}(\mathsf{hk}, x_i) = y_i$ and

$$\delta\big(\pi_i([x_i, y_i] \cdot \mathbf{G} \oplus M_i),\; c'\big) \leq p \quad \text{for } i \in \{1, 2\}.$$

By the triangle inequality,

$$\delta\big(\pi_1([x_1, y_1] \cdot \mathbf{G} \oplus M_1),\; \pi_2([x_2, y_2] \cdot \mathbf{G} \oplus M_2)\big) \;\leq\; 2p.$$

Since $p < 1/4$, we have $2p < 1/2$, so the relation $\mathcal{R}_{\mathbf{G},\pi_1,\pi_2,M_1,M_2,2p}$ from Definition 7.15 is satisfied.

We now construct a reduction $\mathcal{B}$ to the CI security of Π_{Hash}. The reduction $\mathcal{B}$ receives hash key hk from the CI challenger (with security parameter λ_{Hash}), guesses (s_1, s_2), derives $(\pi_1, M_1), (\pi_2, M_2)$, simulates $(\mathsf{pk}_{\mathsf{code}}, \mathsf{sk}_{\mathsf{code}}) \leftarrow \Pi_{\mathsf{code}}.\mathsf{Setup}(1^\lambda)$, and runs $\mathcal{A}$ on $\mathsf{pk} = (\mathsf{pk}_{\mathsf{code}}, \mathsf{hk})$. When $\mathcal{A}$ wins, $\mathcal{B}$ extracts the two pairs $(x_1, y_1), (x_2, y_2)$ from the list L and outputs them to the CI challenger.

Since π_1, π_2, M_1, M_2 are now fixed, Lemma 7.16 applies directly (the sparsity condition $\Omega(v^\alpha) \geq s_0(\lambda_{\mathsf{Hash}})$ holds by our choice of v). The advantage of $\mathcal{B}$ in the CI game is at least $\varepsilon_{\mathcal{A}}/2^{2\lambda}$, and by the CI security of Π_{Hash} (with security parameter λ_{Hash}), this must be negligible in λ_{Hash}. Concretely, by Theorem 3.18, the CI advantage is at most $2^{-\lambda_{\mathsf{Hash}}^{c'}}$. Therefore:

$$\frac{\varepsilon_{\mathcal{A}}}{2^{2\lambda}} \;\leq\; 2^{-\lambda_{\mathsf{Hash}}^{c'}},$$

which gives

$$\varepsilon_{\mathcal{A}} \;\leq\; 2^{2\lambda} \cdot 2^{-\lambda_{\mathsf{Hash}}^{c'}} \;=\; 2^{2\lambda - \lambda_{\mathsf{Hash}}^{c'}}.$$

By our choice of $\lambda_{\mathsf{Hash}} = \lambda^{\lceil 2/c' \rceil}$, we have $\lambda_{\mathsf{Hash}}^{c'} \geq \lambda^2$, so

$$\varepsilon_{\mathcal{A}} \;\leq\; 2^{2\lambda - \lambda^2} \;=\; \mathsf{negl}(\lambda).$$

This completes the proof of unique decoding, and hence of Theorem 7.9. □

7.3 Optimal Rate CCA Codes

CCA Security. While the constructions in the above sections Constructions 7.3 and 7.21 give the optimal rate to robustness tradeoff in against CPA adversaries, we also study the stronger CCA setting, where the adversarial channel is additionally given access to a decoding oracle. In this setting, the work of [AAC+25] gives a generic Fujisaki-Okamoto style transformation from any CPA-secure public-key pseudorandom code to a CCA-secure one in the random oracle model. And relevant for our use case, this transformation preserves the asymptotic rate and robustness of the underlying CPA secure code! So morally, we would like to directly apply this transformation to our construction in Constructions 7.3 and 7.21; however, there are a couple technical gaps:

1. The prior transformation requires the underlying CPA-secure PRC satisfy a stronger notion of *adaptive robustness* - that is, it is hard for an adversary to break robustness even after seeing encoding randomness. Our CPA codes do not satisfy this.

2. The prior transformation does not handle the case of list decoding. In particular, the adversarial channel now receives qualitatively *more* when querying a list-decoding oracle.

The remainder of this section will be devoted to closing these gaps. To address (1), while our codes do not satisfy the stronger notion of adaptive robustness (in particular, an adversary is who sees encoding randomness can efficiently erase the small fraction of bits corresponding to the encoding of the underlying PRG seed), we show in Section 7.3.1 that a simple modification of Construction 7.3 in the random oracle model yields a weaker notion of simulation decoding, where decoding failures (for adversaries who see encoding randomness) can be efficiently simulated without the decoding key. We then show in Section 7.3.3 that this weaker notion is sufficient to utilize the [AAC+25] CPA to CCA transformation. At the same time, we address (2) by extending the analysis of this transformation to handle adversaries with list-decoding oracles.

Uniqueness for Free. Although the goal of this section is achieving CCA security, we show that the same transformation in the random oracle model also converts list-decodable codes to uniquely decodable codes for $p < 1/4$ fraction of errors. In other words, in the random oracle model, we can avoid the techniques from section 7.2 and get CCA security and unique decoding simultaneously via one transformation.

More formally, in this section, we will show the following two lemmas:

Lemma 7.23 (Simulation Decoding in the Random Oracle Model)**.** *Let $p, \varepsilon > 0$ be constants such that $p < 1/2$ and let $\lambda \in \mathbb{N}$ be a security parameter. Suppose there exists a public-key zero-bit CPA code Π_{PRC} that is pseudorandom and $(p+\varepsilon)$-robust. Then in the random oracle model: There exists a polynomial L and a public-key CPA list-decodable pseudorandom and $(p, L(\lambda))$-robust code that achieves rate $R = 1 - H(p) - \varepsilon$, that satisfies the p-simulation decoding property.*

Lemma 7.24 (CCA security from CPA security with simulatable decoding, in the random oracle model)**.** *Let $0 \leq p < 1/2$ be constants such that and let $\lambda \in \mathbb{N}$ be a security parameter. Suppose there exists a public-key CPA list-decodable that is pseudorandom and $(p, L = L(\lambda))$-robust, and has p-simulatable decoding with rate R. Then in the random oracle model: There exists a public-key CCA list-decodable code that is pseudorandom and (p, L')-robust for $L' \leq L$, and achieves rate $R - o(1)$. Moreover, when $p < \frac{1}{4}$, $L' = 1$ and the code is uniquely decodable.*

Combining Lemmas 7.23 and 7.24 gives us Theorem 7.2.

The proof for Lemma 7.23 is given in Section 7.3.1 and the proof for Lemma 7.24 is given in Section 7.3.3.

7.3.1 Achieving Simulation-Decodability

As described in Section 7.1, Construction 7.3 does not satisfy simulation-decodability. This is because the success of the decoding algorithm of Construction 7.3 is heavily dependent on a randomly dispersed but sparse set of codeword positions T, where the PRG seed s is embedded. While a (potential) simulator, given access to an encoding along with its randomness, can determine when an induced error vector erases these locations, the adversary can 'fool' the decoder by re-embedding the PRG seed in a different sparse subset of bits T'. Since this can be done under fresh PRC randomness, the simulator cannot tell if it has been given random sparse noise or an a redundant embedding of s. As such, we need to enable the decoding algorithm to make this check and distinguish these cases. Here can we give a simple modification of the construction in the random oracle model which is provably simulation-decodable.

Construction 7.25. Let $\lambda \in \mathbb{N}$ be a security parameter, and parameters p, α, m, n be set as Construction 7.3. Let Π be the code constructed in Construction 7.3 on message space $k + \lambda$, using ingredients $\Pi_{\mathsf{PRC}}, \Pi_{\mathsf{BSC}}, \mathsf{PRG}$. We will additionally assume the underlying Π_{PRC} is also simulation-decodable. Let H be random oracles with domain $\{0,1\}^{k+\lambda}$.

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: On input security parameter λ, the setup algorithm simply runs the $(\mathsf{pk}_{\mathsf{PRC}}, \mathsf{sk}_{\mathsf{PRC}}) \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{Setup}(1^\lambda)$ and outputs $(\mathsf{pk} = \mathsf{pk}_{\mathsf{PRC}}, \mathsf{sk} = (\mathsf{pk}_{\mathsf{PRC}}, \mathsf{sk}_{\mathsf{PRC}}))$.
- $\mathsf{Enc}(\mathsf{pk}, x; s) \to c$: On input public key $\mathsf{pk} = \mathsf{pk}_{\mathsf{PRC}}$ and message $x \in \{0,1\}^k$, and randomness $s \in \{0,1\}^\lambda$, the encoding algorithms computes the following:
  - Evaluate $(T, V, \{r_i\}_{i \in T}, \{d_i\}_{i \in V}) = H(x, s)$ where $T, V \subseteq [q]$ are uniformly sampled as disjoint subsets of size $|T|, |V| = \alpha/10 \cdot q$, each $r_i \leftarrow \{0,1\}^\lambda$ and $d_i \leftarrow \{0,1\}^m$.

- Compute $(\pi, M) \leftarrow \mathsf{PRG}(s)$, where $\pi : [n] \to [n]$ and $M \in \{0,1\}^n$.
- Compute $c_{\mathsf{BSC}} \leftarrow \Pi_{\mathsf{BSC}}.\mathsf{Enc}(x)$.
- Apply permuting and masking $c' = \pi(M \oplus c_{\mathsf{BSC}})$.
- Divide $c' = c'_1, \ldots, c'_q$ as q blocks of length m.
- Compute and output $c = c_1, \ldots c_q$, where for each $i \in [q]$,

$$c_i = \begin{cases} \Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}, s; r_i) & i \in T \\ d_i & i \in V \\ c'_i & i \notin T \cup V \end{cases}$$

- $\mathsf{ListDec}(\mathsf{sk}, c) \to L$: On input secret key $\mathsf{sk} = (\mathsf{pk}_{\mathsf{PRC}}, \mathsf{sk}_{\mathsf{PRC}})$ and codeword c, the decoding algorithm computes the following:
  - Parse $c = c_1, \ldots, c_q$.
  - For each $i \in [q]$, recover list $L_i \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{ListDec}(\mathsf{sk}_{\mathsf{PRC}}, c_i)$ and define the set:

$$L \;=\; \{\, s : |\{\, i \in [q] : s \in L_i \,\}| \geq |T| \cdot \alpha/5\}.$$

  - For each $s \in L$, derive $(\pi, M) \leftarrow \mathsf{PRG}(s)$ and recover $x \leftarrow \mathsf{BSC}.\mathsf{Dec}(\pi^{-1}(c) \oplus M)$. Recompute $c' = c'_1, \ldots, c'_q \leftarrow \mathsf{Enc}(\mathsf{pk}, x; s)$ and $(T, V, \{r_i\}_{i \in T}, \{d_i\}_{i \in V}) = H(x, s)$, and checks:

$$\left|\left\{i \in T \;:\; \delta(c'_i, c_i) \leq p + \alpha \wedge s \in L_i\right\}\right| \geq |T| \cdot \alpha/5 \text{ and } |\{i \in V \;:\; \delta(d_i, c_i) \leq p + \alpha\}| \geq |V| \cdot \alpha/5 \qquad (7.3)$$

  - Define L_{out} to be the set of x's recovered in the previous step (which passed the check). Output L_{out}.

Lemma 7.26 (Robustness, pseudorandomness, simulated decodability, and rate for Construction 7.25)**.** *Let $0 < p, \alpha$ be constants with $p + \alpha < 1/2$. Assume the ingredients in Construction 7.25. Then the code Π defined in Construction 7.25 is a public-key CPA list-decodable (p, L)-robust, pseudorandom code with p-simulatable decoding, rate $R = k/n$, where $L = c \cdot L_{\mathsf{PRC}}$ and c is a universal constant that depends on α in the random oracle model.*

Proof of Lemma 7.23.

Proof of Lemma 7.23. The proof follows along the same line as the proof of Theorem 7.1, where instead of using Lemma 7.4 we make use of Lemma 7.26. □

We prove Lemma 7.26 in Section 7.3.2 below.

7.3.2 Proof of Lemma 7.26

Rate. We can see that the message and codeword length of this construction is simply that of Construction 7.3. Thus the rate of the code is the same as the previous construction.

Pseudorandomness. Observe that compared to Construction 7.3, the encoding algorithm changes in two ways - (1) the randomness $T, \{r_i\}_{i \in T}$ is derived from the random oracle (rather than freshly sampled by the encoding algorithm), and (2) the random oracle also outputs new blocks d_i to replace some fraction of the codeword. Thus, to argue pseudorandomness, we must argue that the adversary does not query H on challenge query and randomness (x, s). It is overwhelmingly unlikely an adversary does this before receiving the challenge (as their view is independent of s) and if the adversary is able to do query (x, s) with noticeable probability afterwards, then they can break pseudorandomness of Construction 7.3, as before said query, this randomness is only encoded in the challenge codeword (see Claim 7.33 for a formal version of an analogous argument).

Robustness. Robustness follows from a similar argument to Construction 7.3 as well. Observe that ListDec in this construction simply checks an additional condition. Recall we previously argue (per Eq. (7.1)) that on a random set T, the number of such blocks that contain $\leq p + \alpha$ relative errors on the robustness challenge is overwhelmingly likely to be $\geq |T| \cdot \alpha/5$. We can apply this same argument to both sets T, V of our construction, which are random in the view of an adversary which does not query $H(x, s)$. The argument then proceeds exactly as Construction 7.3, albeit with a mildly larger (but still $< p + \alpha$ fraction) error from V.

Simulated Decodability. We will consider the following simulator. We note our simulator achieves a slightly weaker definition, where it makes use of the random oracle queries of the adversary (which we denote $\mathcal{H}$). However, this weaker notion is sufficient for realizing the CPA to CCA transformation Section 7.3, where the decoding simulator for this construction will only be used by the challenger in the proof of security.

- $\mathsf{Sim}^{\mathcal{H}}(\mathsf{pk}, x, s, c) \to \{0, 1\}$: On input public key pk, message x, randomness s and codeword c, and list $\mathcal{H} = \{(x_i, s_i)\}$ of random oracle queries made by $\mathcal{A}$, the decoding simulator:
  - Parse $c = c_1, \ldots, c_q$.
  - For each $(x_j, s_j) \in \mathcal{H}$ where $x_j = x$:
    * Evaluate $(T, V, \{r_i\}_{i \in T}, \{d_i\}_{i \in V}) = H(x_j, s_j)$.
    * Derive $(\pi, M) \leftarrow \mathsf{PRG}(s_j)$ and compute $x' \leftarrow \mathsf{BSC.Dec}(\pi^{-1}(c) \oplus M)$.
    * If $x' = x$, for each $i \in T$, reconstruct $c'_i \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{PRC}}, s_j; r_i)$, checking,

$$\left|\left\{i \in T \ : \ \delta(c'_i, c_i) \leq p + \alpha \wedge \Pi_{\mathsf{PRC}}.\mathsf{Sim}(\mathsf{pk}_{\mathsf{PRC}}, s_j, r_i, c_i) = 1\right\}\right| \geq |T| \cdot \alpha/5$$

$$\text{and } |\{i \in V \ : \ \delta(d_i, c_i) \leq p + \alpha\}| \geq |V| \cdot \alpha/5$$

      If so, output 1.
  - If no such (x_j, s_j) exists, output 0.

To see that this is indistinguishable from the real decoder, consider the following sequence of hybrids:

$\mathsf{ExptDS}^{(0)}$**:** This is the real decoding simulation experiment when $b = 0$ (using the real ListDec).

1. On input parameter λ, the challenger $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and sends pk to the challenger
2. As many times as desired, the adversary makes queries:
   (a) The adversary submits x, s, c.
   (b) The challenger parses $c = c_1, \ldots c_q$.
   (c) For each $i \in [q]$, recover list $L_i \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{ListDec}(\mathsf{sk}_{\mathsf{PRC}}, c_i)$ and define the set:

$$L \ = \ \{ s : |\{ i \in [q] : s \in L_i \}| \geq |T| \cdot \alpha/5\}.$$

   (d) For each $s_j \in L$, derive $(\pi, M) \leftarrow \mathsf{PRG}(s)$ and recover $x_j \leftarrow \mathsf{BSC.Dec}(\pi^{-1}(c) \oplus M)$. If $x_j = x$, recompute $c' = c'_1, \ldots, c'_q \leftarrow \mathsf{Enc}(\mathsf{pk}, x_j; s_j)$ and $(T, V, \{r_i\}_{i \in T}, \{d_i\}_{i \in V}) = H(x_j, s_j)$, and check:

$$\left|\left\{i \in T \ : \ \delta(c'_i, c_i) \leq p + \alpha \wedge s_j \in L_i\right\}\right| \geq |T| \cdot \alpha/5 \text{ and } |\{i \in V \ : \ \delta(d_i, c_i) \leq p + \alpha\}| \geq |V| \cdot \alpha/5$$

      If so, return 1.
   (e) If no such s_j exists, return 0.
3. The adversary outputs a bit b', which is the output of the experiment.

H_0 : In this experiment, the challenger will remove any codewords from L for which (x, s) are not in the random oracle queries $\mathcal{H}$ made by $\mathcal{A}$.

2. As many times as desired, the adversary makes queries:

(d) For each $s_j \in L$, derive $(\pi, M) \leftarrow \mathsf{PRG}(s)$ and recover $x_j \leftarrow \mathsf{BSC.Dec}(\pi^{-1}(c) \oplus M)$. If $x_j = x$ and $(x_j, s_j) \in \mathcal{H}$, recompute $c' = c'_1, \ldots, c'_q \leftarrow \mathsf{Enc}(\mathsf{pk}, x_j; s_j)$ and $(T, V, \{r_i\}_{i\in T}, \{d_i\}_{i\in V}) = H(x_j, s_j)$, and check:

$$\left|\left\{i \in T \ : \ \delta(c'_i, c_i) \le p + \alpha \wedge s_j \in L_i\right\}\right| \ge |T| \cdot \alpha/5 \text{ and } |\{i \in V \ : \ \delta(d_i, c_i) \le p + \alpha\}| \ge |V| \cdot \alpha/5$$

If so, return 1.

Claim 7.27. $\left|\Pr[\mathsf{ExptDS}^{(0)}] - \Pr[H_0]\right| \le \mathsf{negl}(\lambda)$.

Proof. Take any (x_j, s_j) output by $\Pi_{\mathsf{PRC}}.\mathsf{ListDec}$ not contained in $\mathcal{H}$. Consider the terms $V, \{d_i\}_{i\in V}$ output by $H(x_j, s_j)$. Since $\mathcal{A}$ did not query $H(x_j, s_j)$, these are uniform in the view of $\mathcal{A}$, and hence independent of the challenge codeword c'. Thus, for all $i \in V$, $d_i \oplus c_i$ is uniform in $\{0,1\}^m$, and since $p + \alpha$ is a constant $< \frac{1}{2}$, by a simple Chernoff bound, overwhelmingly unlikely to satisfy $\delta(d_i, c_i) \le p + \alpha$. □

Claim 7.28. $\left|\Pr[H_0] - \Pr[\mathsf{ExptDS}^{(1)}]\right| \le \mathsf{negl}(\lambda)$.

Proof. We can reduce this to the simulated decodability of our underlying code Π_{PRC}. Consider the following reduction:

1. $\mathcal{B}$ receives public key $\mathsf{pk}_{\mathsf{PRC}}$ from the challenger, and forwards it to $\mathcal{A}$.
2. As many times as desired, $\mathcal{A}$ makes queries:
   (a) The adversary submits $x \in \{0,1\}^k, s \in \{0,1\}^\lambda, c \in \{0,1\}^n$.
   (b) The reduction $\mathcal{B}$ parses $c = c_1, \ldots c_q$.
   (c) For each $(x_j, s_j) \in \mathcal{H}$, derive $(\pi, M) \leftarrow \mathsf{PRG}(s_j)$ and check that $x_j \leftarrow \mathsf{BSC.Dec}(\pi^{-1}(c) \oplus M)$. If so, compute random oracle $(T, V, \{r_i\}_{i\in T}, \{d_i\}_{i\in V}) = H(x_j, s_j)$ and recompute $c'_i \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{PRC}}, s_j; r_i)$. For all $i \in T$, submit query s_j, r_i, c_i to the challenger. Define set T_j to be all i for which the challenger responds with 1. Check that
   $$\left|\left\{i \in T_j \ : \ \delta(c'_i, c_i) \le p + \alpha\right\}\right| \ge |T| \cdot \alpha/5 \text{ and } |\{i \in V \ : \ \delta(d_i, c_i) \le p + \alpha\}| \ge |V| \cdot \alpha/5$$
   If so, return 1.
   (d) If no such s_j exists, return 0.
3. Adversary $\mathcal{A}$ returns a bit b', which $\mathcal{B}$ simply returns as well.

First, observe that when the challenger is responding with $\Pi_{\mathsf{PRC}}.\mathsf{ListDec}$, then the set T_j corresponds to the set $\{i : s_j \in L_i\}$, where $L_i \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{ListDec}(\mathsf{sk}_{\mathsf{PRC}}, c_i)$ in $\mathsf{ExptDS}^{(0)}, H_0$. Thus we can see

$$\left\{i \in T_j \ : \ \delta(c'_i, c_i) \le p + \alpha\right\} = \left\{i \in T \ : \ \delta(c'_i, c_i) \le p + \alpha \wedge s_j \in L_i\right\},$$

and so $\mathcal{B}$ simulates $\mathcal{A}$'s view of H_0.

On the other hand, when the challenger responds with $\Pi_{\mathsf{PRC}}.\mathsf{Sim}$, the set T_j corresponds to the set $\{i : \Pi_{\mathsf{PRC}}.\mathsf{Sim}(\mathsf{pk}_{\mathsf{PRC}}, s_j, r_i, c_i) = 1\}$, and so

$$\left\{i \in T_j \ : \ \delta(c'_i, c_i) \le p + \alpha\right\} = \left\{i \in T \ : \ \delta(c'_i, c_i) \le p + \alpha \wedge \Pi_{\mathsf{PRC}}.\mathsf{Sim}(\mathsf{pk}_{\mathsf{PRC}}, s_j, r_i, c_i) = 1\right\},$$

meaning $\mathcal{B}$ responds as $\mathsf{Sim}^{\mathcal{H}}$ defined above, simulating $\mathcal{A}$'s view of $\mathsf{ExptDS}^{(1)}$. □

7.3.3 From Public-Key CPA Codes With Simulation Decoding to CCA Codes in the RO Model

In this subsection we prove Lemma 7.24. We revisit the Fujisaki-Okamoto style transform of [AAC+25] for boosting CPA to CCA security for pseudorandom codes in the random oracle model. We show that this same transformation satisfies a few additional properties:

1. It suffices for the underlying Π_{PRC} code to satisfy p-robustness and p-simulatable decoding.
2. If the underlying Π_{PRC} is list-decodable, then the resulting code is CCA list-decodable.
3. When $p < \frac{1}{4}$, the list is of size 1, and hence the resulting code is uniquely decodable.

Construction 7.29. Let λ be the security parameter and let p be a constant $< \frac{1}{2}$. Take the message space k and codeword length n to be $> \lambda$. We make use of the following ingredients.

- Let Π_{PRC} be a list-decodable pseudorandom code on message space $\{0,1\}^{k+2\lambda}$ and codeword space $\{0,1\}^n$ which satisfies p-robustness and p-simulatable decoding.
- Let H be a random oracle from $\{0,1\}^k \times \{0,1\}^\lambda \to \{0,1\}^\lambda \times \{0,1\}^\lambda$.

We now describe the transformation:

- $\mathsf{Setup}(1^\lambda)$: On input security parameter λ, the setup algorithm samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{Setup}(1^\lambda)$. It outputs pk, sk as the public and secret keys respectively. Without loss of generality, we will assume the secret key includes a copy of the public key as well.
- $\mathsf{Enc}(\mathsf{pk}, x)$: On input public key pk and input message $x \in \{0,1\}^k$, the encoding algorithm samples $r \leftarrow \{0,1\}^\lambda$, and evaluates $(y, z) = H(x, r)$ (where $y, z \in \{0,1\}^\lambda$). It then computes $c \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}, (x, r, y); z)$ (where z is used as the encoding randomness).
- $\mathsf{ListDec}(\mathsf{sk}, c')$: On input secret key sk and string c', the decoding algorithm runs $L \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{ListDec}(\mathsf{sk}, c')$. Output
$$\{x \ : \ (x, r, y) \in L \wedge (y, z) = H(x, r) \wedge \delta(c', \Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}, (x, r, y); z)) \leq p\}.$$

Lemma 7.30 (CCA security for list-decodable codes)**.** *Let p be a constant $0 \leq p < \frac{1}{2}$. Suppose Π_{PRC} is a public-key list-decodable CPA code that is pseudorandom, $(p, L = L(\lambda))$-robust, has p-simulatable decoding, and achieves rate R. Then the code Π defined in Construction 7.29 is a public-key list-decodable CCA code that is pseudorandom, (p, L')-robust for $L' \leq L$, and achieves rate $R - o(1)$ in the random oracle model. Moreover, when $p < \frac{1}{4}$, $L' = 1$ and the code is uniquely decodable.*

The proof for Lemma 7.30 is given in Section 7.3.4.

Proof of Lemma 7.24. The proof of Lemma 7.24 follows directly from Lemma 7.30. □

7.3.4 Proof of Lemma 7.30

Rate. We can see that a codeword to the above construction is simply a codeword to Π_{PRC} on a message of length $k + 2\lambda$. Since λ is independent of k, this has asymptotic rate equal to that of Π_{PRC}.

The code is CCA pseudorandom. We will reduce to the pseudorandomness and simulatable decodability of Π_{PRC}. We consider the following sequence of hybrids:

$\mathsf{ExptPR}^{(0)}$**:** This is the real public key pseudorandomness experiment when $b = 0$ with oracle access to decoding oracle (per Definition 3.10) instantiated with Construction 7.29. The challenger lazily samples the random oracle.

1. On input parameter λ, the challenger samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{Setup}(1^\lambda)$, and sends pk to adversary $\mathcal{A}$. The challenger keeps track of mapping $\mathcal{H} : \{0,1\}^k \times \{0,1\}^\lambda \to \{0,1\}^\lambda \times \{0,1\}^\lambda \cup \perp$ (initializing all entries to $\perp$).
2. As many times as desired, the adversary makes queries:
   (a) **Random Oracle Query:** On input (x_i, r_i), if $\mathcal{H}[(x_i, r_i)] = \perp$, sample $y_i, z_i \leftarrow \{0,1\}^\lambda$ and update $\mathcal{H}[(x_i, r_i)] := (y_i, z_i)$. In either case, return $\mathcal{H}[(x_i, r_i)]$.
   (b) **Decoding Query:** On input $c' \in \{0,1\}^n$, sample $L \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{ListDec}(\mathsf{sk}, c')$. Set
   $$L' = \{x \ : \ (x, r, y) \in L \ \wedge \ (y, z) = H(x, r) \wedge \delta(c', \Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}, (x, r, y); z)) \leq p\}.$$
   and return L' to $\mathcal{A}$ (note this induces a random oracle query on each $(x, r) \in L$).
3. The adversary submits a challenge message x^*. The challenger responds as:
   (a) Sample $r^* \leftarrow \{0,1\}^\lambda$ and query $(y^*, z^*) = H(x^*, r^*)$.
   (b) Sample $c^* \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}, (x^*, r^*, y^*); z^*)$, returning c^* to $\mathcal{A}$.
4. The adversary continues making **Random Oracle** and **Decoding** queries.
   (a) **Random Oracle Query:** Exactly as before challenge query.
   (b) **Decoding Query:** On input $c' \in \{0,1\}^n$, compute L' as Step 2.(b). If $\delta(c, c^*) \leq p$, return $L' \cup \{x^*\}$, otherwise, return L'.
5. The adversary $\mathcal{A}$ submits a guess bit b', which is the output of the experiment.

H_0**:** Same as $\mathsf{ExptPR}^{(0)}$, except on decoding queries, the challenger only includes list elements for which the random oracle has previously already been queried on. As a result, decoding queries no longer induce (implicit) random oracle queries.

2. As many times as desired, the adversary makes queries:
   b. **Decoding Query:** On input $c' \in \{0,1\}^n$, sample $L \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{ListDec}(\mathsf{sk}, c')$. Set
   $$L' = \{x \ : \ (x, r, y) \in L \wedge (y, z) = \mathcal{H}[(x, r)] \ \wedge \ \delta(c', \Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}, (x, r, y); z)) \leq p\}.$$
   and return L' to adversary $\mathcal{A}$.

H_1**:** Same as H_0, except the challenger simulates decoding queries.

2. As many times as desired, the adversary makes queries:
   b. **Decoding Query:** On input $c' \in \{0,1\}^n$, compute
   $$L' = \left\{ x \ \middle| \ \begin{array}{c} (y, z) = \mathcal{H}[(x, r)] \\ \wedge \ \delta(c', \Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}, (x, r, y); z)) \leq p \\ \wedge \ \mathsf{Sim}(\mathsf{pk}, ((x, r, y), z), c') = 1 \end{array} \right\}.$$
   and return L' to adversary $\mathcal{A}$.

H_2**:** Same as H_1, except the challenger aborts with output 0 if the adversary queries H on (x^*, r^*).

3. The adversary submits a challenge on message x^*. The challenger responds as:
   a. Sample $r^* \leftarrow \{0,1\}^\lambda$. If $\mathcal{H}[(x^*, r^*)] \neq \perp$, abort with output 0. Otherwise, query $(y^*, z^*) = H(x^*, r^*)$.
4. The adversary continues making **Random Oracle** and **Decoding** queries.
   a. **Random Oracle Query:** On input (x, r), if $(x, r) = (x^*, r^*)$, abort with output 0. Otherwise, answer as before.

H_3**:** Same as H_2, except the challenge codeword c^* is generated as with fresh encoding randomness. Note that the challenge query no longer induces a random oracle query on (x^*, r^*).

3. The adversary submits a challenge message x^*. The challenger responds as:
   a. Sample $r^* \leftarrow \{0,1\}^\lambda$. If $\mathcal{H}[(x^*, r^*)] \neq \bot$, abort with output 0.
   b. Sample $y^* \leftarrow \{0,1\}^\lambda$. Compute $c^* \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}, (x^*, r^*, y^*))$, returning c^* to $\mathcal{A}$.

H_4**:** Same as H_3, except the challenge codeword c^* is as a uniformly random string.

3. The adversary submits a challenge message x^*. The challenger responds as:
   b. Sample $c^* \leftarrow \{0,1\}^n$, returning c^* to $\mathcal{A}$.

Claim 7.31. $\left|\Pr[\mathsf{ExptPR}^{(0)}] - \Pr[H_0]\right| \leq \mathsf{negl}(\lambda)$.

Proof. Let c' be the input to an arbitrary decoding query, take any $(x, r, y) \in L$ where for which (x, r) has not been queried on H. Recall in $\mathsf{ExptPR}^{(0)}$, the random oracle output is then sampled $y', z' \leftarrow \{0,1\}^\lambda$. Since y' is uniform and independent of the prior view of $\mathcal{A}$ (and hence c'), it is only equal to y (and hence in the output list) with negligible probability. □

Claim 7.32. $|\Pr[H_0] - \Pr[H_1]| \leq \mathsf{negl}(\lambda)$.

Proof. This follows from a reduction to simulatable decodability of Π_{PRC}.

1. $\mathcal{B}$ receives public key pk from the challenger, from which it sends to $\mathcal{A}$. It initializes mapping $\mathcal{H}$ with entries $\bot$.

2. As many times as desired, $\mathcal{A}$ makes queries:

   (a) **Random Oracle Query:** On input (x_i, r_i), if $\mathcal{H}[(x_i, r_i)] = \bot$, sample $y_i, z_i \leftarrow \{0,1\}^\lambda$ and update $\mathcal{H}[(x_i, r_i)] := (y_i, z_i)$. In either case, return $\mathcal{H}[y]$.

   (b) **Decoding Query:** On input $c' \in \{0,1\}^n$, $\mathcal{B}$ initializes empty list $L' = \varnothing$ and computes the following:

      i. For each non-$\bot$ entry $\mathcal{H}[(x_i, r_i)] = (y_i, z_i)$, compute $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}, (x_i, r_i, y_i); z_i)$.
      ii. If $\delta(c', c_i) \leq p$, $\mathcal{B}$ submits challenge query message (x_i, r_i, y_i), randomness z_i and error $c' \oplus c_i$ to the challenger.
      iii. If the challenger's response is 1, append $L' := L' \cup \{x_i\}$.

   $\mathcal{B}$ returns L' to $\mathcal{A}$.

3. Reduction $\mathcal{B}$ computes challenge message honestly,

4. $\mathcal{A}$ continues to make **Random Oracle** and **Decoding** queries, which $\mathcal{B}$ answers as before (except appending challenge x^* to the decoding response when $\delta(c', c^*) \leq p$).

5. $\mathcal{A}$ returns a bit, which is the output of the experiment.

Observe that when the challenger's response is the indicator for $(x, r, y) \in L \leftarrow \mathsf{ListDec}(\mathsf{sk}, c')$, then $\mathcal{B}$ is exactly the computing L' as H_0. When the challenger's response is the simulated decoding, this likewise corresponds exactly to H_1. □

Claim 7.33. $|\Pr[H_1] - \Pr[H_2]| \leq \mathsf{negl}(\lambda)$.

Proof. First, observe that since r^* is uniform over $\{0,1\}^\lambda$, since any efficient $\mathcal{A}$ only makes a polynomial number of queries to H, the probability $\mathcal{A}$ makes such a query before receiving the challenge is always negligible.

Now suppose $\mathcal{A}$ makes such a random oracle query after receiving the challenge. Consider the following reduction $\mathcal{B}$ to the pseudorandomness of Π_{PRC}:

1. $\mathcal{B}$ receives public key pk from the challenger, from which it sends to $\mathcal{A}$. It initializes mapping $\mathcal{H}$ as in $\mathsf{ExptPR}^{(0)}$.

2. As many times as desired, $\mathcal{A}$ makes queries:

(a) **Random Oracle Query:** On input (x_i, r_i), if $\mathcal{H}[(x_i, r_i)] = \bot$, sample $y_i, z_i \leftarrow \{0,1\}^\lambda$ and update $\mathcal{H}[(x_i, r_i)] := (y_i, z_i)$. In either case, return $\mathcal{H}[(x_i, r_i)]$.

(b) **Decoding Query:** On input $c' \in \{0,1\}^n$, compute

$$L' = \left\{ x \;\middle|\; \begin{array}{c} (y,z) = \mathcal{H}[(x,r)] \\ \wedge\, \delta(c', \Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}, (x,r,y); z)) \leq p \\ \wedge\, \mathsf{Sim}(\mathsf{pk}, ((x,r,y),z), c') = 1 \end{array} \right\}.$$

to $\mathcal{A}$.

3. Reduction $\mathcal{B}$ receives challenge message x^* from $\mathcal{A}$. They then sample $r^*, y^* \leftarrow \{0,1\}^\lambda$, and submits message (x^*, r^*, y^*) to the challenger, receiving challenge codeword c^*, which is forwarded to $\mathcal{A}$.

4. $\mathcal{A}$ continues to make **Random Oracle** and **Decoding** queries, which $\mathcal{B}$ answers as H_1. If a random oracle query is made on (x^*, r^*), abort with output 1.

5. If $\mathcal{A}$ terminates, $\mathcal{B}$ outputs 0.

First, note that when $\mathcal{B}$ receives a random challenge $c^* \leftarrow \{0,1\}^n$, then the distribution of $\mathcal{A}$ is independent of r^*. Thus we can conclude that $\mathcal{A}$ will query (x^*, r^*) with negligible probability, and so with overwhelming probability $\mathcal{B}$ will output 0.

On the other hand, observe that when $\mathcal{B}$ receives $c^* \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}, (x^*, r^*, y^*))$, the challenge is distributed exactly as H_0 (implicitly setting the challenge randomness to z^*). Thus, the only difference between the state of the H_0 challenger and the reduction $\mathcal{B}$ is that the value of $\mathcal{H}[(x^*, r^*)] = \bot$ in the latter. Now we can examine the queries following the challenge.

a. **Random Oracle Query:** On any query $(x, r) \neq (x^*, r^*)$, the response of $\mathcal{B}$ and H_0 are identical.

b. **Decoding Query:** On decoding query c', we can consider a couple cases:

- Decoding list L does not contain an element of the form (x^*, r^*, y), then the outputs of H_0, $\mathcal{B}$ are identical.
- L contains a elements of the form (x^*, r^*, y) and $\delta(c', c^*) \leq p$. In this case, x^* is explicitly hard-coded as part of the decoding output by both $\mathcal{B}$ and H_0 challenger.
- L contains a elements of the form (x^*, r^*, y) and $\delta(c', c^*) > p$. In this case, the H_0 challenger will recover (y^*, z^*) from $\mathcal{H}$, and recover challenge codeword $c^* \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{Enc}(pk, (x^*, r^*, y^*); z^*)$. However, since $\delta(c', c^*) > p$, this element will be removed from the output of the decoding query. On the other hand, $\mathcal{B}$ will remove (x^*, r^*, y) from the query output as $\mathcal{H}[(x^*, r^*)] = \bot$ in $\mathcal{B}$.

In all three cases, the outputs of $\mathcal{B}$ and the H_0 challenger are identical.

Thus, the view of $\mathcal{A}$ is identical up until it attempts to query $H(x^*, r^*)$, at which point $\mathcal{B}$ aborts with output 1. □

Claim 7.34. $|\Pr[H_2] - \Pr[H_3]| \leq \mathsf{negl}(\lambda)$.

Proof. Since in H_1, the adversary never queries the random oracle on (x^*, r^*) with overwhelming probability, the view of the adversary is independent of (y^*, z^*), so this hybrid is identical. □

Claim 7.35. $|\Pr[H_3] - \Pr[H_4]| \leq \mathsf{negl}(\lambda)$.

Proof. Since c^* is now sampled on independent randomness, this follows from directly from pseudorandomness of Π_{PRC}. □

Claim 7.36. $\left|\Pr[H_4] - \Pr[\mathsf{ExptPR}^{(1)}]\right| \leq \mathsf{negl}(\lambda)$.

Proof. Since the codeword is now random, we can now revert the changes of H_0, H_1, H_2, which brings us to $\mathsf{ExptPR}^{(1)}$. □

Putting together these claims directly gives us the indistinguishability of $\mathsf{ExptPR}^{(0)}, \mathsf{ExptPR}^{(1)}$, showing our scheme is pseudorandom.

The code is CCA p-robust, and uniquely decodable when $p < \frac{1}{4}$. We will simultaneously argue that this code is p-robust with respect to a decoding oracle, and when constant $p < \frac{1}{4}$, is uniquely decodable. To do this, we will consider the following experiment which outputs 1 if the adversary queries the decoding oracle on an input c' such that either (1) The distance to challenge codeword $\delta(c', c^*) \leq p$ but x^* is not in the output of the decoding query, or (2) The decoding query contains > 1 message (when $p < \frac{1}{4}$). We will move to an experiment where both are impossible. Note that in the CCA setting, without loss of generality we can assume any adversary to robustness can query the decoding oracle on their final response. This argument will rely on the pseudorandomness, robustness, and simulatable decodability of Π_{PRC}.

ExptAR**:** This is the real public key p-robustness experiment with oracle access to decoding (per Definition 3.10) instantiated with Construction 7.29.

1. On input parameter λ, the challenger samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{Setup}(1^\lambda)$, and sends pk to adversary $\mathcal{A}$. The challenger keeps track of mapping $\mathcal{H} : \{0,1\}^k \times \{0,1\}^\lambda \to \{0,1\}^\lambda \times \{0,1\}^\lambda \cup \perp$ (initializing all entries to $\perp$).
2. As many times as desired, the adversary makes queries:
   (a) **Random Oracle Query:** On input (x_i, r_i), if $\mathcal{H}[(x_i, r_i)] = \perp$, sample $y_i, z_i \leftarrow \{0,1\}^\lambda$ and update $\mathcal{H}[(x_i, r_i)] := (y_i, z_i)$. In either case, return $\mathcal{H}[(x_i, r_i)]$.
   (b) **Decoding Query:** On input $c' \in \{0,1\}^n$, sample $L \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{ListDec}(\mathsf{sk}, c')$. Set
   $$L' = \{x \ : \ (x, r, y) \in L \ \wedge \ (y, z) = H(x, r) \ \wedge \ \delta(c', \Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}, (x, r, y); z)) \leq p\}.$$
   and return L' to $\mathcal{A}$ (note this induces a random oracle query on each $(x, r) \in L$).
3. The adversary submits a challenge message x^*. The challenger responds as:
   (a) Sample $r^* \leftarrow \{0,1\}^\lambda$ and query $(y^*, z^*) = H(x^*, r^*)$.
   (b) Sample $c^* \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}, (x^*, r^*, y^*); z^*)$, returning c^* to $\mathcal{A}$.
4. The adversary continues making **Random Oracle** and **Decoding** queries.
   (a) **Random Oracle Query:** Exactly as before challenge query.
   (b) **Decoding Query:** On input $c' \in \{0,1\}^n$, compute and return L' as Step 2.(b).
      i. If $\delta(c^*, c') \leq p$ and $x^* \notin L'$, abort with output 1.
      ii. If $p < \frac{1}{4}$ and $|L'| > 1$, abort with output 1.
5. If $\mathcal{A}$ terminates, the experiment ends with output 0.

H_0**:** Same as ExptAR, except on decoding queries, the challenger only includes list elements for which the random oracle has previously already been queried on. As a result, decoding queries no longer induce (implicit) random oracle queries.

2. As many times as desired, the adversary makes queries:
   b. **Decoding Query:** On input $c' \in \{0,1\}^n$, sample $L \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{ListDec}(\mathsf{sk}, c')$. Set
   $$L' = \{x \ : \ (x, r, y) \in L \wedge (y, z) = \mathcal{H}[(x, r)] \ \wedge \ \delta(c', \Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}, (x, r, y); z)) \leq p\}.$$
   and return L' to adversary $\mathcal{A}$.

H_1**:** Same as H_0, except the challenger simulates decoding queries on non-challenge codewords.

2. As many times as desired, the adversary makes queries:

b. **Decoding Query:** On input $c' \in \{0,1\}^n$, compute

$$L' = \left\{ x \;\middle|\; \begin{array}{c} (y,z) = \mathcal{H}[(x,r)] \\ \wedge\ \delta(c', \Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}, (x,r,y); z)) \leq p \\ \wedge\ \mathsf{Sim}(\mathsf{pk}, ((x,r,y),z), c') = 1 \end{array} \right\}.$$

and return L' to adversary $\mathcal{A}$.

4. The adversary continues making **Random Oracle** and **Decoding** queries.
   b. **Decoding Query:** On input $c' \in \{0,1\}^n$, compute L' as Step 2.(b).
      i. If $\delta(c^*, c') > p$, return L'. If $\delta(c^*, c') \leq p$, compute $L \leftarrow \mathsf{ListDec}(\mathsf{sk}, c')$. If $x^* \notin L$, abort with output 1. Otherwise, return $L' \cup \{x^*\}$.

H_2**:** Same as H_1, except the challenger aborts with output 0 if the adversary queries H on (x^*, r^*).

3. The adversary submits a challenge on message x^*. The challenger responds as:
   a. Sample $r^* \leftarrow \{0,1\}^\lambda$. If $\mathcal{H}[(x^*, r^*)] \neq \perp$, abort with output 0. Otherwise, query $(y^*, z^*) = H(x^*, r^*)$.

H_3**:** Same as H_2, except the challenger aborts if it queries the random oracle on the challenge codeword, and hard codes the challenge codeword in decoding queries.

4. The adversary continues making **Random Oracle** and **Decoding** queries.
   a. **Random Oracle Query:** On input (x, r), if $(x, r) = (x^*, r^*)$, abort with output 0. Otherwise, answer as before.
   b. **Decoding Query:** On input $c' \in \{0,1\}^n$, compute L' as Step 2.(b). If $\delta(c^*, c') \leq p$, return $\{x^*\} \cup L'$. Otherwise, return L'.

H_4**:** Same as H_3, except when $p < \frac{1}{4}$, each time the random oracle is queried, the challenger aborts with output 0 if the resulting codeword is distance $1 - 2 \cdot p$ from a prior codeword.

2. As many times as desired, the adversary makes queries:
   a. **Random Oracle Query:** On input (x_i, r_i), if $\mathcal{H}[(x_i, r_i)] = \perp$, sample $y_i, z_i \leftarrow \{0,1\}^\lambda$. Otherwise set $\mathcal{H}[(x_i, r_i)] = (y_i, z_i)$ and return $\mathcal{H}[(x_i, r_i)]$.
      i. When $p < \frac{1}{4}$, compute $c_i \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}, (x_i, r_i, y_i); z_i)$. If for some $j < i$, $\delta(c_j, c_i) \leq 2 \cdot p$, abort with output 0.
3. The adversary submits a challenge on message x^*. The challenger responds as:
   b. Sample $c^* \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}, (x^*, r^*, y^*); z^*)$, returning c^* to $\mathcal{A}$.
      i. Let $|\mathsf{RO}|$ be the number of random oracle queries made so far. When $p < \frac{1}{4}$, if for some $i \in [|\mathsf{RO}|]$ $\delta(c^*, c_i) \leq 2 \cdot p$, abort with output 0. For indexing purposes, let $c^* = c_{|\mathsf{RO}|+1}$ and include this in the count of random oracle queries.

Claim 7.37. $|\Pr[\mathsf{ExptAR}] - \Pr[H_0]| \leq \mathsf{negl}(\lambda)$.

Proof. This argument is identical to Claim 7.31. □

Claim 7.38. $|\Pr[H_0] - \Pr[H_1]| \leq \mathsf{negl}(\lambda)$.

Proof. This reduction is identical to Claim 7.32. Note, however since unlike the pseudorandomness game, the decoding oracle is not punctured on the challenge codeword in the robustness game, the challenger still explicitly list decodes for challenge message x^*. □

Claim 7.39. $|\Pr[H_1] - \Pr[H_2]| \leq \mathsf{negl}(\lambda)$.

Proof. Since r^* is uniform over $\{0,1\}^\lambda$, since any efficient $\mathcal{A}$ only makes a polynomial number of queries to H, the probability $\mathcal{A}$ makes such a query before receiving the challenge is always negligible. □

Claim 7.40. $|\Pr[H_2] - \Pr[H_3]| \leq \mathsf{negl}(\lambda)$.

Proof. To argue this, we will define a series of subhybrids ranging over the queries the adversary has made.

H_{2,i^*}**:** Same as H_2, except on the first i^* queries after the challenge, answer as H_3.

Claim 7.41. $\left|\Pr[H_{2,i^*-1}] - \Pr[H_{2,i^*}]\right| \leq \mathsf{negl}(\lambda)$.

Suppose an adversary $\mathcal{A}$ has noticeable difference in advantage between H_{2,i^*-1} and H_{2,i^*}. Then one of two events must occur with non-negligible probability

Event E_1**:** Query i^* was the first random oracle query on (x^*, r^*), meaning H_{2,i^*} aborts with output 0.

Event E_2**:** Query i^* was a decoding query on c' such that $\delta(c^*, c') \leq p$ and $x^* \notin \mathsf{ListDec}(\mathsf{sk}, c')$, meaning H_{2,i^*-1} aborts with output 0.

Event E_1 occurs with negligible probability We can argue this via the pseudorandomness of Π_{PRC}, in an argument similar to Claim 7.33. Note here we rely on the fact that in both hybrids, any decoding queries in the first $i^* - 1$ no longer make use of ListDec. Consider reduction $\mathcal{B}$ as follows:

1. $\mathcal{B}$ receives public key pk from the challenger, from which it sends to $\mathcal{A}$. It initializes mapping $\mathcal{H}$.
2. As many times as desired, $\mathcal{A}$ makes queries:
   (a) **Random Oracle Query:** On input (x_i, r_i), if $\mathcal{H}[(x_i, r_i)] = \bot$, sample $y_i, z_i \leftarrow \{0,1\}^\lambda$ and update $\mathcal{H}[(x_i, r_i)] := (y_i, z_i)$. In either case, return $\mathcal{H}[(x_i, r_i)]$.
   (b) **Decoding Query:** On input $c' \in \{0,1\}^n$, compute
   $$L' = \left\{ x \;\middle|\; \begin{array}{c} (y,z) = \mathcal{H}[(x,r)] \\ \wedge\, \delta(c', \Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}, (x,r,y); z)) \leq p \\ \wedge\, \mathsf{Sim}(\mathsf{pk}, ((x,r,y),z), c') = 1 \end{array} \right\}.$$
   to $\mathcal{A}$.
3. Reduction $\mathcal{B}$ receives challenge message x^* from $\mathcal{A}$. They then sample $r^*, y^* \leftarrow \{0,1\}^\lambda$, and submits message (x^*, r^*, y^*) to the challenger, receiving challenge codeword c^*, which is forwarded to $\mathcal{A}$.
4. $\mathcal{A}$ continues to make **Random Oracle** and **Decoding** queries. For queries $i < i^*$:
   (a) **Random Oracle Query:** Continue answering queries as Step 2.(a). If $(x, r) = (x^*, r^*)$, $\mathcal{B}$ aborts and outputs 0.
   (b) **Decoding Query:** On input $c' \in \{0,1\}^n$, compute L' as Step 2.(b). If $\delta(c^*, c') \leq p$, return $\{x\} \cup L'$. Otherwise, return L'.
5. On query i^*, $\mathcal{A}$ submits a **Random Oracle** query to (x^*, r^*), $\mathcal{B}$ outputs 1. Otherwise, $\mathcal{B}$ outputs 0.

First, observe when $\mathcal{B}$ receives $c^* \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}, (x^*, r^*, y^*))$, the reduction exactly simulates adversary $\mathcal{A}$'s view in H_{2,i^*-1}, H_{2,i^*} up until the query i^*. Then we can see that $\mathcal{B}$ only outputs 1 when event E_1 occurs. On the other hand, when $c^* \leftarrow \{0,1\}^n$, r^* is information-theoretically hidden from the view of the adversary, and so the probability of querying on (x^*, r^*) must be negligible.

Event E_2 occurs with negligible probability This relies on the p-robustness of Π_{PRC}. Note that here, we rely on the fact that the random oracle was not queried on (x^*, r^*) before query i^* in both hybrids, meaning the challenge is sampled on fresh randomness in the view of the adversary. Consider reduction $\mathcal{B}$ as follows:

1. $\mathcal{B}$ receives public key pk from the challenger, from which it sends to $\mathcal{A}$. It initializes mapping $\mathcal{H}$.
2. As many times as desired, $\mathcal{A}$ makes queries:
   (a) **Random Oracle Query:** On input (x_i, r_i), if $\mathcal{H}[(x_i, r_i)] = \bot$, sample $y_i, z_i \leftarrow \{0,1\}^\lambda$ and update $\mathcal{H}[(x_i, r_i)] := (y_i, z_i)$. In either case, return $\mathcal{H}[(x_i, r_i)]$.
   (b) **Decoding Query:** On input $c' \in \{0,1\}^n$, compute
   $$L' = \left\{ x \;\middle|\; \begin{array}{c} (y,z) = \mathcal{H}[(x,r)] \\ \wedge\, \delta(c', \Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}, (x,r,y); z)) \le p \\ \wedge\, \mathsf{Sim}(\mathsf{pk}, ((x,r,y),z), c') = 1 \end{array} \right\}.$$
   to $\mathcal{A}$.
3. Reduction $\mathcal{B}$ receives challenge message x^* from $\mathcal{A}$. They then sample $r^*, y^* \leftarrow \{0,1\}^\lambda$, and submits message (x^*, r^*, y^*) to the challenger, receiving challenge codeword c^*, which is forwarded to $\mathcal{A}$.
4. $\mathcal{A}$ continues to make **Random Oracle** and **Decoding** queries. For queries $i < i^*$:
   (a) **Random Oracle Query:** Continue answering queries as Step 2.(a). If $(x, r) = (x^*, r^*)$, $\mathcal{B}$ aborts and outputs 0.
   (b) **Decoding Query:** On input $c' \in \{0,1\}^n$, compute L' as Step 2.(b). If $\delta(c^*, c') \le p$, return $\{x\} \cup L'$. Otherwise, return L'.
5. On query i^*, $\mathcal{A}$ submits a **Decoding** query on string c', $\mathcal{B}$ forwards c' to the robustness challenger.

Like the previous case, we can observe that $\mathcal{B}$ exactly simulates adversary $\mathcal{A}$'s view in H_{2,i^*-1}, H_{2,i^*} up until the query i^*. Moreover, since $\mathcal{A}$ does not query the random oracle on (x^*, r^*) (without the experiment aborting), the randomness z^* to the challenge does not need to be simulated. Then, observe that event E_2 corresponds exactly to $\mathcal{B}$ winning the p-robustness game.

Since $\mathcal{A}$ only makes a polynomial number of queries, Claim 7.41 suffices to show Claim 7.40. □

Claim 7.42. $|\Pr[H_3] - \Pr[H_4]| \le \mathsf{negl}(\lambda)$.

Proof. We argue this via pseudorandomness of Π_{PRC}. Much like Claim 7.40, we define a sequence of sub-experiments indexed by query:

H_{3,i^*}**:** Same as H_3, except on the i^* random oracle queries, perform the random oracle check as H_4. For simplicity of indexing, we will include the challenger's random oracle query when computing the challenge codeword as well.

Claim 7.43. $\left|\Pr[H_{3,i^*-1}] - \Pr[H_{3,i^*}]\right| \le \mathsf{negl}(\lambda)$.

Proof. Observe that an adversary $\mathcal{A}$ which has wins in H_{3,i^*-1} but not H_{3,i^*}, on the i^{*th} random oracle query, $\mathcal{A}$ mus have queried a point (x_{i^*}, r_{i^*}) such that the resulting codeword $c_{i^*} \leftarrow \Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}, (x_i, r_i, y_i); z_i)$ is p-close to a previous codeword. Call this event E_{i^*}. Using that adversary, consider the following reduction $\mathcal{B}$ against pseudorandomness of Π_{PRC}.

1. $\mathcal{B}$ receives public key pk from the challenger, which it sends to $\mathcal{A}$. It initializes empty mapping $\mathcal{H}$.
2. As many times as desired, $\mathcal{A}$ makes queries. On query $i < i^*$.
   (a) **Random Oracle Query:** $\mathcal{B}$ answers as H_{3,i^*-1}, saving c_i as the codeword.

(b) **Decoding Query:** $\mathcal{B}$ answers as H_{3,i^*-1}.

3. $\mathcal{B}$ computes the challenge codeword exactly as a **Random Oracle**, sampling $r^* \leftarrow \{0,1\}^\lambda$ rather than receiving it from the adversary.

4. Continue answering queries until the i^{*th} random oracle query.

5. On the i^{*th} random oracle query on input (x_{i^*}, r_{i^*}), sample $y_{i^*} \leftarrow \{0,1\}^\lambda$ and submit $(x_{i^*}, r_{i^*}, y_{i^*})$ to the pseudorandomness challenger. On response codeword c_{i^*}, check that for all $i < i^*$, $\delta(c_i, c_{i^*}) \leq 2 \cdot p$. It so, abort with output 0. Otherwise, abort with output 1. In either case, the reduction $\mathcal{B}$ terminates.

First, observe by construction that $\mathcal{B}$ perfectly simulates the experiment H_{3,i^*-1}, H_{3,i^*} up until the i^{*th} query. In addition, when query i^* is c_{i^*} is computed as $\Pi_{\mathsf{PRC}}.\mathsf{Enc}(\mathsf{pk}, (x_{i^*}, r_{i^*}, y_{i^*}))$, then $\mathcal{B}$ return 1 exactly when E_{i^*} occurs.

Now, observe that when $c_{i^*} \leftarrow \{0,1\}^n$, for any fixed string c_i, we can concentration bound the probability that $\delta(c_i, c_{i^*}) \leq (2 \cdot p)$. Recall p is a constant $< \frac{1}{4}$, so $2 \cdot p$ is a constant less than $\frac{1}{2}$. Write $\varepsilon = \frac{1}{2} - 2 \cdot p > 0$. Taking each the probability each bit of c_i, c_{i^*} agree as uniform independent expectation $\frac{1}{2}$ random variables, we apply multiplicative Chernoff to bound $\Pr[\delta(c_i, c_{i^*}) \leq \frac{1}{2} - \varepsilon] \leq e^{-\varepsilon^2 \cdot n/2}$. Since ε is a constant and $n > \lambda$, this is negligible, even when union bounding over all polynomially many c_i. Thus, $\mathcal{B}$ returns 1 with negligible probability.

Combining these two observations means that the advantage of $\mathcal{B}$ in the pseudorandomness game is $E_{i^*} - \mathsf{negl}(\lambda)$, so E_{i^*} must occur with negligible probability. □

Since $\mathcal{A}$ only makes a polynomial number of queries, Claim 7.43 suffices to show Claim 7.42. □

In our final hybrid, we can observe that from the changes of H_3 that the adversary can no longer give a decoding query which breaks robustness, and from H_4, each pair of codewords c_0, c_1 derived from the random oracle has $\delta(c_0, c_1) \geq 2 \cdot p$ for $p < \frac{1}{4}$. Thus, no two such codewords can lie in a radius p-hamming ball. Since by this hybrid decoding queries only return messages derived from the random oracle within a radius p hamming ball, each decoding query returns at most 1 message.

8 Acknowledgements

The third author's research was supported by NSF grants CNS-2349972 and CNS-2055510. Research supported in part by a grant from the UC Noyce Initiative to the Simons Institute for the Theory of Computing. Part of this work was conducted while the authors were visiting the Simons Institute for the Theory of Computing.

References

[AAC+25] Omar Alrabiah, Prabhanjan Ananth, Miranda Christ, Yevgeniy Dodis, and Sam Gunn. Ideal pseudorandom codes. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, pages 1638–1647, 2025.

[BHMW21] Elette Boyle, Justin Holmgren, Fermi Ma, and Mor Weiss. On the security of doubly efficient PIR. *IACR Cryptol. ePrint Arch.*, page 1113, 2021.

[BW21] Keller Blackwell and Mary Wootters. A note on the permuted puzzles toy conjecture. *CoRR*, abs/2108.07885, 2021.

[BZ76] EL Blokh and VV Zyablov. Generalized concatenated codes. *Svyaz', Moscow*, 1976.

[CG24] Miranda Christ and Sam Gunn. Pseudorandom error-correcting codes. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VI*, volume 14925 of *LNCS*, pages 325–347, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[CGG+26] Miranda Christ, Noah Golowich, Sam Gunn, Ankur Moitra, and Daniel Wichs. Improved pseudorandom codes from permuted puzzles, 2026.

[Eli57] Peter Elias. List decoding for noisy channels. 1957.

[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, Berkeley, CA, USA, August 11-13, 2025*, volume 353 of *LIPIcs*, pages 54:1–54:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025.

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

[GI01] Venkatesan Guruswami and Piotr Indyk. Expander-based constructions of efficiently decodable codes. In *42nd FOCS*, pages 658–667, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press.

[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.

[GS98] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. In *39th FOCS*, pages 28–39, Palo Alto, CA, USA, November 8–11, 1998. IEEE Computer Society Press.

[GS99] V. Guruswami and M. Sudan. Improved decoding of reed-solomon and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999.

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

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

[JLLW23] Aayush Jain, Huijia Lin, Ji Luo, and Daniel Wichs. The pseudorandom oracle model and ideal obfuscation. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 233–262, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland.

[KSS19] S. Kopparty, R. Shaltiel, and J. Silbak. Quasilinear time list-decodable codes for space bounded channels. *To appear in the 60th Annual Symposium on Foundations of Computer Science (FOCS)*, 2019.

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

[LSW25] George Lu, Jad Silbak, and Daniel Wichs. Binary codes for computationally bounded errors under standard crypto assumptions. In *FOCS*, 2025.

[LV22] Alex Lombardi and Vinod Vaikuntanathan. Correlation-intractable hash functions via shift-hiding. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 102:1–102:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

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

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

[Smi07] A. D. Smith. Scrambling adversarial errors using few random bits, optimal information reconciliation, and better private codes. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 395–404, 2007.

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

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

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

[SS24a] R. Shaltiel and J. Silbak. Explicit codes for poly-size circuits and functions that are hard to sample on low entropy distributions. *Electronic Colloquium on Computational Complexity (ECCC)*, 2024.

[SS24b] Ronen Shaltiel and Jad Silbak. Explicit codes for poly-size circuits and functions that are hard to sample on low entropy distributions. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *56th ACM STOC*, pages 2028–2038, Vancouver, BC, Canada, June 24–28, 2024. ACM Press.

[Sud97] M. Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13, 1997.

[SW25a] Jad Silbak and Daniel Wichs. Binary codes for error detection and correction in a computationally bounded world. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part I*, volume 15601 of *Lecture Notes in Computer Science*, pages 186–211. Springer, 2025.

[SW25b] Jad Silbak and Daniel Wichs. Detecting and correcting computationally bounded errors: A simple construction under minimal assumptions. In Raghu Meka, editor, *16th Innovations in Theoretical Computer Science Conference, ITCS 2025, Columbia University, New York, NY, USA, January 7-10, 2025*, volume 325 of *LIPIcs*, pages 88:1–88:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025.

[Woz58] John M. Wozencraft. List decoding. 1958.

[Zya71] Victor V. Zyablov. An estimate of the complexity of constructing binary linear cascade codes. *Problemy Peredachi Informatsii*, 7(1):5–13, 1971. In Russian. Also cited as "On Estimation of Complexity of Construction of Binary Linear Concatenated Codes." English translation: *Problems of Information Transmission*, 7(1):3–10, 1971.

Appendix

A Shannon-Rate Linear Codes with Pairwise Column-Robustness

We start by defining linear codes and dual codes.

Definition A.1 (Linear codes and dual codes)**.** Let $\mathbb{F}_q$ denote the field with q element. An $[n,k]_q$ **linear code** is a linear subspace of $C \subseteq \mathbb{F}_q^n$ of dimension k. We say that C has distance d, if the Hamming weight of every nonzero vector in C is at least d. Such codes are called $[n,k,d]_q$ codes. A linear map $\mathsf{Enc} : \mathbb{F}_q^k \to \mathbb{F}_q^n$ is an **encoding function** for C, if $\mathsf{Enc}(\mathbb{F}_q^k) = C$. We say that $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ is the generating matrix of C, if for ever $m \in \mathbb{F}_q^k$: $\mathsf{Enc}(m) = m \cdot \mathbf{G}$. For a code C, we use $C^\perp$ to denote the dual vector space. We say that C has **dual distance** d if $C^\perp$ has distance d.

We will make use of the following theorem for binary inner codes that simultaneously achieve the Shannon rate for the binary symmetric channel and have constant dual distance. This follows from a standard probabilistic construction over a bounded-independence family of binary linear codes: with high probability, a code in the family has exponentially small maximum-likelihood decoding error on $\mathrm{BSC}(p)$ and dual distance at least 3. Since the family has size $\mathrm{poly}(2^b)$, such a code can also be found in time $\mathrm{poly}(2^b)$ by exhaustive search.

Theorem A.2 (BSC-good inner codes with dual distance at least three)**.** *Fix constants $p \in (0, 1/2)$ and $\varepsilon > 0$, and let $R = 1 - H(p) - \varepsilon$. For all sufficiently large block lengths b, there exists a binary linear code $C \subseteq \{0,1\}^b$ of dimension $\ell = \lfloor Rb \rfloor$ such that:*

1. *C can be decoded from $\mathrm{BSC}(p)$ errors, by maximum-likelihood decoding, with error probability at most $2^{-\Omega(b)}$;*
2. *the code has dual distance at least 3.*

Moreover, such a code can be found in time $\mathrm{poly}(2^b)$ by exhaustive search.

We will use the following claim which will help us reason about the linear independence of the columns of the generating matrix of a code.

Claim A.3. *Let $\mathsf{Enc} : \mathbb{F}_q^k \to \mathbb{F}_q^n$ be an encoding function for a linear $[n,k]_q$-code C with a generating matrix $\mathbf{G} \in \mathbb{F}_q^{k \times n}$. Applying Enc on a uniformly chosen message $m \leftarrow \mathbb{F}_q^k$ (i.e., $\mathsf{Enc}(m) = m \cdot \mathbf{G}$) yields a distribution $(Z_1, \ldots, Z_n)$ over $\mathbb{F}_q^n$. For a subset $T \subseteq [n]$, it holds that random variables $(Z_i)_{i \in T}$ are uniform and independent iff $\mathsf{rank}([\mathbf{G}]_T) = |T|$.*

Proof. Follows directly by linearity. □

t-Wise Independent. We say that Random variables $R_1, \ldots, R_n$ are t-wise independent if for every $i_1, \ldots, i_t \in [n]$, $R_{i_1}, \ldots, R_{i_t}$ are uniform and independent.

By the above claim, it is easy to derive the the following standard fact that encoding functions for codes with dual distance $t+1$ yield a t-wise independent distributions.

Lemma A.4 (t-wise independence from linear codes with $t+1$ dual-distance)**.** *Let $\mathsf{Enc} : \mathbb{F}_q^k \to \mathbb{F}_q^n$ be an encoding function for a linear $[n,k]_q$-code C with dual distance $t+1$. Applying Enc on a uniformly chosen message $m \leftarrow \mathbb{F}_q^k$ yields a distribution $(Z_1, \ldots, Z_n)$ over $\mathbb{F}_q^n$ that is t-wise independent, and every Z_i is uniformly distributed over $\mathbb{F}_q$.*

Proof. Applying Enc on some $v \in \mathbb{F}^k$, can be seen as multiplying v by a generator matrix of C (which is a transposed parity check matrix of $C^\perp$). As $C^\perp$ has distance $t+1$, every t columns of the generator matrix of C are linearly independent, and the lemma follows. □

We now define code Concatenation. Looking ahead, the structure of the code will help us argue the column-robustness that we need.

Definition A.5 (Concatenated code)**.** Given functions

- $\mathsf{Enc}_{\mathsf{out}} : \{0,1\}^{k_{\mathsf{out}}} \to (\{0,1\}^{k_{\mathsf{in}}})^{n_{\mathsf{out}}}$, and
- $\mathsf{Enc}_{\mathsf{in}} : \{0,1\}^{k_{\mathsf{in}}} \to (\{0,1\}^{\log q_{\mathsf{in}}})^{n_{\mathsf{in}}}$,

we define the *concatenated encoding function* $\mathsf{Enc} : \{0,1\}^{k_{\mathsf{out}}} \to (\{0,1\}^{\log q_{\mathsf{in}}})^{n_{\mathsf{out}} \cdot n_{\mathsf{in}}}$ denoted by $\mathsf{Enc}_{\mathsf{out}} \circ \mathsf{Enc}_{\mathsf{in}}$ as follows: For $m \in \{0,1\}^{k_{\mathsf{out}}}$, let $\mathsf{Enc}_{\mathsf{out}}(m) = (s_1, \ldots, s_{n_{\mathsf{out}}})$, and $\mathsf{Enc}(m) = \big(\mathsf{Enc}_{\mathsf{in}}(s_1), \mathsf{Enc}_{\mathsf{in}}(s_2), \ldots, \mathsf{Enc}_{\mathsf{in}}(s_{n_{\mathsf{out}}})\big)$.

For $i \in [n_{\mathsf{out}} \cdot n_{\mathsf{in}}]$, let $\mathsf{out}(i) \in [n_{\mathsf{out}}]$, and $\mathsf{in}(i) \in [n_{\mathsf{in}}]$ be the values defined so that

$$i = (\mathsf{out}(i) - 1) \cdot n_{\mathsf{in}} + \mathsf{in}(i),$$

and note that $\mathsf{Enc}(m)_i = \mathsf{Enc}_{\mathsf{in}}\big(\mathsf{Enc}_{\mathsf{out}}(m)_{\mathsf{out}(i)}\big)_{\mathsf{in}(i)}$.

Lemma A.6. *Let* $\mathsf{Enc}_{\mathsf{out}} : \mathbb{F}_q^k \to \mathbb{F}_q^{n_{\mathsf{out}}}$ *be an encoding function for a linear* $[n_{\mathsf{out}}, k]_q$*-code* C_{out}*, and let* $\mathsf{Enc}_{\mathsf{in}} : \mathbb{F}_2^{\log q} \to \mathbb{F}_2^{n_{\mathsf{in}}}$ *be an encoding function for a linear* $[n_{\mathsf{in}}, \log(q)]_2$*-code* C_{in}*. Let* $\mathbf{G}$ *be the generating matrix for the linear code* $[n \coloneqq n_{\mathsf{in}} \cdot n_{\mathsf{out}},\ v \coloneqq \log(q) \cdot k]_2$ *defined by the code concatenation* $\mathsf{Enc} = \mathsf{Enc}_{\mathsf{out}} \circ \mathsf{Enc}_{\mathsf{in}}$*. Assuming*

- C_{out} *has dual-distance* $t + 1$*, and*
- C_{in} *has dual-distance* 3*,*

then for every subset $T \subseteq [n]$*, if* $|T| \le t$*, and for every* $j \in [n_{\mathsf{out}}]$ *it holds that* $\big|i \in T : \mathsf{out}(i) = j\big| \le 2$*, (that is, each outer-code symbol contains at most two indices) it follows that* $\mathsf{rank}([\mathbf{G}]_T) = |T|$.

Proof of Lemma A.6. Fix a subset $T \subseteq [n]$ satisfying $|T| \le t$ and such that for every $j \in [n_{\mathsf{out}}]$,

$$\big|\{\, i \in T : \mathsf{out}(i) = j \,\}\big| \le 2.$$

Let $m \in \mathbb{F}_2^v$ be uniformly random, and let

$$(Z_1, \ldots, Z_n) = \mathsf{Enc}(m) \in \mathbb{F}_2^n.$$

Since $\mathbf{G}$ is a generating matrix of the concatenated code, by Claim A.3 it suffices to show that the random variables $(Z_i)_{i \in T}$ are uniform and independent. Write

$$(W_1, \ldots, W_{n_{\mathsf{out}}}) = \mathsf{Enc}_{\mathsf{out}}(m),$$

where each $W_j \in \mathbb{F}_q = \mathbb{F}_2^{\log q}$. Since C_{out} has dual distance at least $t + 1$, it follows by Lemma A.4 that $(W_1, \ldots, W_{n_{\mathsf{out}}})$ are t-wise independent. In particular, since $|T| \le t$, the collection of outer symbols

$$\big(W_{\mathsf{out}(i)}\big)_{i \in T}$$

is jointly distributed as follows: values coming from distinct outer indices are independent, and each W_j is uniform over $\mathbb{F}_2^{\log q}$.

Let $\mathbf{G}_{\mathsf{in}}$ be a generator matrix of C_{in}, and for each $\ell \in [n_{\mathsf{in}}]$ let $c_\ell \in \mathbb{F}_2^{\log q}$ denote the ℓ-th column of $\mathbf{G}_{\mathsf{in}}$. By definition of concatenation, for every $i \in T$,

$$Z_i = \mathsf{Enc}(m)_i = \mathsf{Enc}_{\mathsf{in}}\big(\mathsf{Enc}_{\mathsf{out}}(m)_{\mathsf{out}(i)}\big)_{\mathsf{in}(i)} = \mathsf{Enc}_{\mathsf{in}}(W_{\mathsf{out}(i)})_{\mathsf{in}(i)} = \langle W_{\mathsf{out}(i)}, c_{\mathsf{in}(i)} \rangle.$$

Now fix $j \in [n_{\mathsf{out}}]$, and let $T_j \coloneqq \{\, i \in T : \mathsf{out}(i) = j \,\}$ and note that by assumption $|T_j| \le 2$.

If $T_j = \varnothing$, there is nothing to prove. If $T_j = \{i\}$, then

$$Z_i = \langle W_j, c_{\mathsf{in}(i)} \rangle.$$

Since C_{in} has dual distance 3, no column of $\mathbf{G}_{\mathsf{in}}$ is zero, so $c_{\mathsf{in}(i)} \neq 0$. Therefore, because W_j is uniform over $\mathbb{F}_2^{\log q}$, the bit Z_i is uniform over $\mathbb{F}_2$.

If $T_j = \{i, i'\}$ with $i \neq i'$, then

$$(Z_i, Z_{i'}) = \big(\langle W_j, c_{\mathsf{in}(i)}\rangle, \langle W_j, c_{\mathsf{in}(i')}\rangle\big).$$

Since C_{in} has dual distance 3, any two columns of $\mathbf{G}_{\mathsf{in}}$ are linearly independent. Over $\mathbb{F}_2$, this means that $c_{\mathsf{in}(i)}$ and $c_{\mathsf{in}(i')}$ are distinct nonzero vectors, hence linearly independent. It follows that the linear map

$$x \mapsto \big(\langle x, c_{\mathsf{in}(i)}\rangle, \langle x, c_{\mathsf{in}(i')}\rangle\big)$$

from $\mathbb{F}_2^{\log q}$ to $\mathbb{F}_2^2$ has rank 2. Therefore, since W_j is uniform over $\mathbb{F}_2^{\log q}$, the pair $(Z_i, Z_{i'})$ is uniform over $\mathbb{F}_2^2$. In particular, Z_i and $Z_{i'}$ are uniform and independent.

Thus, for each $j \in [n_{\mathsf{out}}]$, the subfamily $(Z_i)_{i\in T_j}$ is uniform and independent. Moreover, if $j \neq j'$, then $(Z_i)_{i\in T_j}$ depends only on W_j, while $(Z_i)_{i\in T_{j'}}$ depends only on $W_{j'}$; and the family of distinct outer symbols $(W_j)_{j:T_j\neq\varnothing}$ is independent because $|T| \leq t$ and C_{out} is $(t+1)$-dual-distance. Hence the full family $(Z_i)_{i\in T}$ is uniform and independent.

Therefore, by Claim A.3,

$$\mathsf{rank}([\mathbf{G}]_T) = |T|.$$

□

Lemma A.7 (Bottom rows of a concatenated RS code). *Let $q = 2^\ell$, and let $\mathsf{Enc}_{\mathsf{out}} : \mathbb{F}_q^k \to \mathbb{F}_q^{n_{\mathsf{out}}}$ and $\mathsf{Enc}_{\mathsf{in}} : \mathbb{F}_2^\ell \to \mathbb{F}_2^{n_{\mathsf{in}}}$ be encoding functions for $\mathrm{RS}_q(n_{\mathsf{out}}, k)$ and an $[n_{\mathsf{in}}, \ell]_2$-code C_{in}, respectively. Let $\mathsf{Enc} = \mathsf{Enc}_{\mathsf{out}} \circ \mathsf{Enc}_{\mathsf{in}} : \mathbb{F}_2^{k\ell} \to \mathbb{F}_2^{n_{\mathsf{out}}n_{\mathsf{in}}}$ be the concatenated code, and let $\mathbf{G} \in \mathbb{F}_2^{k\ell\times n_{\mathsf{out}}n_{\mathsf{in}}}$ be a generator matrix for Enc. There is an ordering to the rows of $\mathbf{G}$ such that for $t = s\ell$ when considering the matrix $\mathbf{B} \in \mathbb{F}_2^{t\times n_{\mathsf{out}}n_{\mathsf{in}}}$ defined to be the last t rows of $\mathbf{G}$ it holds that $\mathbf{B}$ is a generating matrix for $\mathsf{Enc}' = \mathsf{Enc}'_{\mathsf{out}} \circ \mathsf{Enc}_{\mathsf{in}} : \mathbb{F}_2^{s\ell} \to \mathbb{F}_2^{n_{\mathsf{out}}n_{\mathsf{in}}}$ where $\mathsf{Enc}'_{\mathsf{out}}$ is $\mathrm{RS}_q(n_{\mathsf{out}}, s) \subseteq \mathrm{RS}_q(n_{\mathsf{out}}, k)$.*

Proof of Lemma A.7. Let $\mathbf{G}_{\mathsf{out}} \in \mathbb{F}_q^{k\times n_{\mathsf{out}}}$ be a generator matrix for $\mathrm{RS}_q(n_{\mathsf{out}}, k)$ whose last s rows generate the subcode $\mathrm{RS}_q(n_{\mathsf{out}}, s) \subseteq \mathrm{RS}_q(n_{\mathsf{out}}, k)$. Such an ordering exists since $\mathrm{RS}_q(n_{\mathsf{out}}, s)$ is a subcode of dimension s.

Let $\mathbf{G}_{\mathsf{in}} \in \mathbb{F}_2^{\ell\times n_{\mathsf{in}}}$ be a generator matrix for $\mathsf{Enc}_{\mathsf{in}}$. Under the fixed identification $\mathbb{F}_q \cong \mathbb{F}_2^\ell$, the concatenated code $\mathsf{Enc} = \mathsf{Enc}_{\mathsf{out}} \circ \mathsf{Enc}_{\mathsf{in}}$ has a generator matrix $\mathbf{G}$ obtained by replacing each entry of $\mathbf{G}_{\mathsf{out}}$ by the corresponding ℓ-bit block and then concatenating with $\mathbf{G}_{\mathsf{in}}$. Equivalently, the ℓ rows of $\mathbf{G}$ arising from each row of $\mathbf{G}_{\mathsf{out}}$ span the concatenation of the corresponding outer-code row with $\mathsf{Enc}_{\mathsf{in}}$.

Therefore, if we order the rows of $\mathbf{G}$ according to the rows of $\mathbf{G}_{\mathsf{out}}$, the last

$$t = s\ell$$

rows of $\mathbf{G}$ are precisely the rows induced by the last s rows of $\mathbf{G}_{\mathsf{out}}$, namely by the subcode $\mathrm{RS}_q(n_{\mathsf{out}}, s)$. Hence the submatrix

$$\mathbf{B} \in \mathbb{F}_2^{t\times n_{\mathsf{out}}n_{\mathsf{in}}}$$

consisting of the last t rows of $\mathbf{G}$ is a generator matrix for the concatenated code

$$\mathsf{Enc}' = \mathsf{Enc}'_{\mathsf{out}} \circ \mathsf{Enc}_{\mathsf{in}} : \mathbb{F}_2^{s\ell} \to \mathbb{F}_2^{n_{\mathsf{out}}n_{\mathsf{in}}},$$

where $\mathsf{Enc}'_{\mathsf{out}}$ is an encoding function for $\mathrm{RS}_q(n_{\mathsf{out}}, s)$. □

Proof of Theorem 7.14. We are now ready to prove Theorem 7.14. We instantiate the decomposable code as a concatenated code whose outer code is a Reed–Solomon code and whose inner code is the BSC-good code with dual distance at least 3 from Theorem A.2. The Reed–Solomon structure lets us isolate the "hash part" $\mathbf{B}$ as the generator matrix of a lower-dimensional concatenated code (Lemma A.7), and the dual-distance properties of the two component codes give us pairwise column-robustness via Lemma A.6.

Proof of Theorem 7.14. Fix the constant $0 \leq p < 1/2$ and a sufficiently small constant $\varepsilon > 0$, and let $R = 1 - H(p) - \varepsilon$.

The code. Let $\mathsf{Enc}_{\mathsf{in}} : \mathbb{F}_2^\ell \to \mathbb{F}_2^b$ be the encoding function of the binary linear inner code C_{in} guaranteed by Theorem A.2, where $\ell = \lfloor Rb \rfloor$, the code C_{in} has dual distance at least 3, and it decodes from $\mathrm{BSC}(p)$ errors with failure probability $2^{-\Omega(b)}$. We take the block length $b = \Theta(\log n)$ to be *logarithmic* in the total codeword length n, chosen large enough

that $\ell = \lfloor Rb \rfloor \geq \log n$; this is the standard choice in the Forney concatenation underlying Theorem 3.4 and ensures the inner code can be found in time $\mathrm{poly}(2^b) = \mathrm{poly}(n)$ by exhaustive search. Set $q = 2^\ell$, so that $q = 2^{\lfloor Rb \rfloor} \geq 2^{\log n} = n$.

Let $\mathsf{Enc}_{\mathsf{out}} : \mathbb{F}_q^K \to \mathbb{F}_q^{n_{\mathsf{out}}}$ be the encoding function of the Reed–Solomon code $\mathrm{RS}_q(n_{\mathsf{out}}, K)$, where $n_{\mathsf{out}} = n/b \leq n \leq q$ (so the alphabet is large enough for a length-n_{out} Reed–Solomon code) and K is chosen so that $K\ell = k + v$ (we assume for simplicity that $\ell \mid (k+v)$; otherwise pad the message length by at most $\ell - 1 = O(\log n)$ bits, which does not affect the rate or the analysis). We take the decomposable code to be the concatenation

$$\mathsf{Enc} = \mathsf{Enc}_{\mathsf{out}} \circ \mathsf{Enc}_{\mathsf{in}} : \mathbb{F}_2^{k+v} \to \mathbb{F}_2^n, \qquad n = b \cdot n_{\mathsf{out}},$$

with generator matrix $\mathbf{G} \in \mathbb{F}_2^{(k+v)\times n}$ (see Definition A.5). We order the rows of $\mathbf{G}$ as in Lemma A.7 and set $\mathbf{B}$ to be the last v rows and $\mathbf{A}$ to be the first k rows, so that $\mathbf{G} = \left[\begin{smallmatrix} \mathbf{A} \\ \mathbf{B} \end{smallmatrix}\right]$.

Rate. The outer Reed–Solomon code has rate K/n_{out}, which we can take to be $1 - \varepsilon$ (so that $\mathsf{Enc}_{\mathsf{out}}$ has relative distance ε), and the inner code has rate $\ell/b \geq R - o(1) = 1 - H(p) - \varepsilon - o(1)$. Hence the concatenated code has rate

$$\frac{k+v}{n} = \frac{K\ell}{n_{\mathsf{out}} \cdot b} = \frac{K}{n_{\mathsf{out}}} \cdot \frac{\ell}{b} \geq (1-\varepsilon)\big(1 - H(p) - \varepsilon\big) \geq 1 - H(p) - O(\varepsilon),$$

which is at least $1 - H(p) - \varepsilon'$ for any desired $\varepsilon' > 0$ by taking ε small enough.

Decomposability. Set $s := v/\ell$, so that $v = s\ell$. By Lemma A.7, with the rows of $\mathbf{G}$ ordered so that the last s rows of the outer generator matrix span the subcode $\mathrm{RS}_q(n_{\mathsf{out}}, s) \subseteq \mathrm{RS}_q(n_{\mathsf{out}}, K)$, the submatrix $\mathbf{B}$ consisting of the last $v = s\ell$ rows of $\mathbf{G}$ is itself a generator matrix of the concatenated code

$$\mathsf{Enc}' = \mathsf{Enc}'_{\mathsf{out}} \circ \mathsf{Enc}_{\mathsf{in}} : \mathbb{F}_2^{s\ell} \to \mathbb{F}_2^n, \qquad \mathsf{Enc}'_{\mathsf{out}} \text{ an encoder for } \mathrm{RS}_q(n_{\mathsf{out}}, s).$$

In particular, $\mathbf{G}$ has full rank and decomposes as required by Definition 7.11. Since $n = \mathrm{poly}(k)$ we have $\ell = \Theta(\log n) = \Theta(\log k)$, so the constraint $\log^5(k) < v < k^{0.49}$ ensures $s = v/\ell = \Omega(\log^4 k) \geq 1$ and $s < v < n_{\mathsf{out}}$; hence $\mathrm{RS}_q(n_{\mathsf{out}}, s)$ is well defined.

Pairwise Column-Robustness. Fix a constant α with $0 < \alpha < 1$, set $w = \lceil v^\alpha \rceil$, and fix any pair of permutations $\pi_1, \pi_2 \in \mathsf{Perm}(n)$. Let $S \subseteq [n]$ be a uniformly random subset of size w, and set $S^* = \pi_1^{-1}(S) \cup \pi_2^{-1}(S)$, so that $|S^*| \leq 2w$. We must show that $\mathrm{rank}([\mathbf{B}]_{S^*}) = |S^*|$ with probability at least $1/q(k, v)$ for some polynomial q.

Recall that $\mathbf{B}$ is the generator matrix of $\mathsf{Enc}' = \mathsf{Enc}'_{\mathsf{out}} \circ \mathsf{Enc}_{\mathsf{in}}$, where:

- $\mathsf{Enc}'_{\mathsf{out}}$ encodes $\mathrm{RS}_q(n_{\mathsf{out}}, s)$, whose dual is the generalized Reed–Solomon code $\mathrm{RS}_q(n_{\mathsf{out}}, s)^\perp$ of dimension $n_{\mathsf{out}} - s$ and minimum distance $s + 1$; hence $\mathsf{Enc}'_{\mathsf{out}}$ has dual distance $s + 1$.
- $\mathsf{Enc}_{\mathsf{in}}$ encodes C_{in}, which has dual distance at least 3.

We apply Lemma A.6 with $t = s$. The lemma requires two conditions on $T := S^*$: that $|T| \leq t = s$, and that every outer block contains at most two indices of T, i.e. for every $j \in [n_{\mathsf{out}}]$,

$$\big|\{\, i \in S^* : \mathsf{out}(i) = j \,\}\big| \leq 2.$$

We verify each condition holds (the second only with inverse-polynomial probability).

Size condition. We have $s = v/\ell$ with $\ell = \Theta(\log n)$, while $w = \lceil v^\alpha \rceil$. Since $\alpha < 1$ and $\ell = O(\log n) = O(\log v + \log n_{\mathsf{out}})$ grows only polylogarithmically, for all sufficiently large v we have $2w = 2\lceil v^\alpha \rceil \leq v/\ell = s$ (here we use $v < k^{0.49}$ so that $\log n = O(\log k)$ and hence $v^\alpha \cdot \ell = o(v)$). Thus $|S^*| \leq 2w \leq s = t$.

At most two per block. The n coordinates are partitioned into n_{out} outer blocks of size $b = \Theta(\log n)$ each (block j consists of the indices i with $\mathsf{out}(i) = j$). We bound the probability that some block contains three or more indices of S^*. Consider any three distinct positions i_1, i_2, i_3 lying in a common block. For each $a \in \{1, 2, 3\}$, the event $i_a \in S^*$ requires $\pi_1(i_a) \in S$ or $\pi_2(i_a) \in S$. A single value $z \in S$ lies in the candidate set $\{\pi_1(i_a), \pi_2(i_a)\}_a$ for at most one position via π_1 (namely $\pi_1^{-1}(z)$) and at most one via π_2 (namely $\pi_2^{-1}(z)$), so each element of S can "cover" at most two of the three positions. Hence the event $\{i_1, i_2, i_3\} \subseteq S^*$ requires at least two distinct elements of the (at most six)

candidate values $\{\pi_b(i_a)\}_{a\in[3],b\in\{1,2\}}$ to lie in S. Since S is a uniformly random set of size w, for any two fixed values the probability both lie in S is at most $(w/n)^2$, so by a union bound over the at most $\binom{6}{2}$ pairs,

$$\Pr_S\left[\{i_1,i_2,i_3\}\subseteq S^*\right]\le\binom{6}{2}\left(\frac{w}{n}\right)^2=O\left(\frac{w^2}{n^2}\right).$$

Union-bounding over all n_{out} blocks and all $\binom{b}{3}=O(\log^3 n)$ triples within each block,

$$\Pr_S\left[\exists\, j:|\{i\in S^*:\text{out}(i)=j\}|\ge 3\right]\le n_{\text{out}}\cdot\binom{b}{3}\cdot O\left(\frac{w^2}{n^2}\right)=O\left(\frac{w^2\log^3 n}{n}\right),$$

using $n_{\text{out}}=n/b\le n$. Since $w=v^\alpha<v<k^{0.49}$ and $n\ge k+v>k$, we have $w^2\log^3 n/n<k^{0.98}\log^3 n/k=k^{-0.02}\log^3 n=o(1)$. Hence with probability at least $1-o(1)\ge 1/2$, both conditions of Lemma A.6 hold.

Whenever both conditions hold, Lemma A.6 gives $\mathsf{rank}([\mathbf{B}]_{S^*})=|S^*|$. Therefore

$$\Pr_S\left[\mathsf{rank}([\mathbf{B}]_{S^*})=|S^*|\right]\ge\tfrac{1}{2},$$

so $\mathbf{B}$ is (q,α)-pairwise well-formed with $q(k,v)\equiv 2$. As the construction is explicit (deterministic), this establishes that the code is α-pairwise column-robust per Definition 7.12.

Efficient Decodability from Permuted Errors. The concatenation of an outer Reed–Solomon code of relative distance ε with the BSC-good inner code C_{in} is exactly the code of Theorem 3.4; hence (Enc,Dec) decodes from permuted errors of relative weight p with probability $1-2^{-\Omega(n^{0.1})}$. □