Black-Box Crypto is Useless for Pseudorandom Codes

Sanjam Garg* Sam Gunn† Mingyuan Wang‡

June 2025

Abstract

A *pseudorandom code* is a keyed error-correction scheme with the property that any polynomial number of encodings appear random to any computationally bounded adversary. We show that the pseudorandomness of any code tolerating a constant rate of random errors cannot be based on black-box reductions to almost any generic cryptographic primitive: for instance, anything that can be built from random oracles, generic multilinear groups, and virtual black-box obfuscation. Our result is *optimal*, as Ghentiyala and Guruswami (2024) observed that pseudorandom codes tolerating any sub-constant rate of random errors exist using a black-box reduction from one-way functions.

The key technical ingredient in our proof is the hypercontractivity theorem for Boolean functions, which we use to prove our impossibility in the random oracle model. It turns out that this easily extends to an impossibility in the presence of "crypto oracles," a notion recently introduced—and shown to be capable of implementing all the primitives mentioned above—by Lin, Mook, and Wichs (EUROCRYPT 2025).

*UC Berkeley and Exponential Science Foundation. Email: sanjamg@berkeley.edu. Author is supported in part by the AFOSR Award FA9550-24-1-0156 and research grants from the Bakar Fund, Supra Inc., and the Stellar Development Foundation.

†UC Berkeley. Email: gunn@berkeley.edu.

‡NYU, Shanghai. Email: mingyuan.wang@nyu.edu.

Contents

1 Introduction **3**
- 1.1 Our contribution . . . 3
- 1.2 Concurrent work . . . 4

2 Technical overview **4**
- 2.1 Definitions . . . 4
- 2.2 Low-noise pseudorandom codes from pseudorandom functions . . . 5
- 2.3 The proof strategy . . . 6
- 2.4 Bounding the error of the compiler . . . 7
- 2.5 Upgrading to a crypto oracle . . . 8

3 Toolkit **9**
- 3.1 Notation . . . 9
- 3.2 Information theory . . . 9
- 3.3 Pseudorandom codes . . . 10
- 3.4 Boolean analysis and hypercontractivity . . . 12
- 3.5 Crypto oracles . . . 13

4 The impossibility **14**
- 4.1 Compiling out the random oracle . . . 14
- 4.2 The main theorem . . . 17

References **18**

1 Introduction

A *pseudorandom code* (PRC) is a keyed error-correction scheme whose codewords appear pseudorandom to any computationally bounded adversary who doesn't know the key. In other words, it is a secret-key pseudorandom encryption scheme with the additional requirement that the decoding algorithm functions even if the transmission is corrupted. Unless otherwise specified, a PRC should be robust to a *constant rate* of random errors.

PRCs were defined by [CG24], where they were shown to be equivalent to robust and undetectable watermarking for large language models. These ideas have since been used for practical watermarking of AI-generated images and videos [GZS25, YZC+25, HLLL25].

Unfortunately, despite several constructions having been proposed [CG24, GG24, GM24], every known PRC either suffers from quasipolynomial-time distinguishing attacks or relies on ad-hoc hardness assumptions. It is therefore natural to ask,

is there a fundamental barrier to constructing a pseudorandom code from generic cryptographic primitives (such as one-way functions, public-key encryption, etc.)?

1.1 Our contribution

We answer this question in the affirmative by showing that the black-box use of virtually all cryptographic primitives cannot yield a PRC resilient to a constant fraction of errors.

Informal Theorem 1. *Relative to a random oracle, there do not exist statistically-secure pseudorandom codes tolerating any constant rate of random bit-flip errors.*[1]

There is a simple construction of pseudorandom codes tolerating any sub-constant error rate in the information-theoretic random oracle model [GG24]. Therefore, our result is essentially optimal in terms of the error rate.

In fact, this theorem actually implies much stronger separation results. In the secret-key setting, the encoder and decoder effectively share a *secret random oracle* which is not accessible to the pseudorandomness adversary.[2] The ideas of [LMW25] enable us to use this fact to prove our stronger separation.

That work introduced *crypto oracles* to prove a strong black-box impossibility result for doubly efficient private information retrieval. A crypto oracle (refer to Section 2.5 or Section 3.5) is a stateless algorithm $\mathcal{B}^{\mathcal{R}}$ with access to a secret random oracle $\mathcal{R}$. The work of [LMW25] showed that crypto oracles can implement effectively all primitives in cryptography, including idealized primitives such as virtual black-box obfuscation and generic multilinear groups.

Informal Corollary 2. *Relative to any crypto oracle, there do not exist statistically secure pseudorandom codes tolerating any constant rate of random bit-flip errors.*

It is therefore not possible to build a PRC using black-box reductions to almost any generic cryptographic primitive.

[1] Random errors are the weakest error model considered in the literature. Since we are proving a lower bound, this only makes our result stronger.

[2] Note that the encoder and decoder can prepend all of their queries to the random oracle $\mathcal{R}$ with the shared secret key sk. Then $\mathcal{R}(\mathsf{sk}\|\cdot)$ is the secret random oracle.

By standard techniques in black-box separation literature [IR89, RTV04], our result immediately yields a black-box separation between secret-key PRCs tolerating a constant rate of errors and any primitives implied by crypto oracles.[3]

Informal Corollary 3. *Pseudorandom codes resilient to a constant rate of random errors are black-box separated from all primitives implied by crypto oracles, including random oracles, virtual black-box obfuscation, and generic multilinear groups.*

1.2 Concurrent work

An independent and concurrent work also demonstrates a black-box impossibility result for pseudorandom codes [DMR25]. Our works use similar arguments involving hypercontractivity to prove the impossibility in the random oracle model. But whereas we go on to prove our impossibility relative to any crypto oracle, they introduce an interesting new class of oracles they call *local oracles* and prove their impossibility relative to this class. A local oracle is, very roughly, one for which most outputs are unrelated to each other. As far as we know, these two oracle classes are incomparable to each other.

2 Technical overview

If the reader is already familiar with the prior work on pseudorandom codes, it should be possible to start at Section 2.3.

We begin this overview by recalling the definition of a pseudorandom code (PRC) and introducing useful notation in Section 2.1. For a more complete discussion of PRC definitions, see, e.g. [CG24, GM24, GG24, AAC+25]. We then reproduce an argument of Ghentiyala and Guruswami [GG24] showing that there exist pseudorandom codes for any o(1) rate of random errors, assuming just the existence of one-way functions.

Our impossibility proof is outlined in Sections 2.3 to 2.5. In Section 2.3 we explain our proof strategy, which is to start by ruling out constructions in the random oracle model. We show how to compile out the random oracle from any PRC, yielding a statistically-secure PRC in the standard model—something that is easily seen to be impossible. We outline our analysis of this compiler in Section 2.4, which is the key technical component of our proof. Finally, we explain how this extends to a black-box separation from most of cryptography in Section 2.5, using an idea of Lin, Mook, and Wichs [LMW25].

2.1 Definitions

In this work, we are interested in pseudorandom codes with robustness to the binary symmetric channel, which introduces random bit-flips. Following the convention in Boolean analysis, we denote this channel by $\mathcal{N}_\rho$. For $\rho \in [0,1)$ and $x \in \{0,1\}^n$, $\mathcal{N}_\rho(x)$ replaces each bit of x with a random bit with probability $1-\rho$. Note that $\mathcal{N}_\rho$ is the binary symmetric channel $\mathsf{BSC}_{(1-\rho)/2}$. This is the weakest model of noise considered in the literature for pseudorandom codes.

[3] Technically, our theorem shows that pseudorandom codes resilient to a constant rate of random errors do not exist relative to any crypto oracle $\mathcal{B}^{\mathcal{R}}$ and a PSPACE oracle. Similar to prior results, this corollary is established by observing that, relative to $\mathcal{B}^{\mathcal{R}}$ and PSPACE, for any primitive $\mathcal{P}$ implied by $\mathcal{B}^{\mathcal{R}}$, $\mathcal{P}$ exists but PRCs do not exist.

For the purposes of our overview, a *pseudorandom code* (or PRC) for $\mathcal{N}_\rho$ will be a pair of polynomial-time randomized algorithms $\mathsf{Enc}, \mathsf{Dec}$ that satisfy the following properties:

- **Completeness / robustness**: If $\mathsf{sk} \leftarrow \{0,1\}^\lambda$ is random, $c \leftarrow \mathsf{Enc}(\mathsf{sk})$, and $\tilde{c} \leftarrow \mathcal{N}_\rho(c)$, then $\mathsf{Dec}(\mathsf{sk}, c) = 1$ with probability $1 - \mathsf{negl}(\lambda)$.
- **Soundness**: For any fixed $x \in \{0,1\}^n$, we have $\Pr_{\mathsf{sk}\leftarrow\{0,1\}^\lambda}[\mathsf{Dec}(\mathsf{sk}, x) = \perp] \geqslant 1 - \mathsf{negl}(\lambda)$.
- **Pseudorandomness**: For any polynomial-time adversary $\mathcal{A}$ and any $m = \mathsf{poly}(\lambda)$,

$$\left| \Pr_{\substack{c_1,\ldots,c_m\leftarrow\mathsf{Enc}(\mathsf{sk}) \\ \mathsf{sk}\leftarrow\{0,1\}^\lambda}}[\mathcal{A}(c_1,\ldots,c_m) = 1] - \Pr_{x_1,\ldots,x_m\leftarrow\{0,1\}^n}[\mathcal{A}(x_1,\ldots,x_m) = 1] \right| \leqslant \mathsf{negl}(\lambda).$$

This definition is only the *secret-key, zero-bit* case of the more general definition of [CG24]. Since we are proving an impossibility, it only strengthens our result to consider this special case.

2.2 Low-noise pseudorandom codes from pseudorandom functions

Before we get to our impossibility, let us recall how pseudorandom codes for $\mathcal{N}_\rho$ can be built from pseudorandom functions (PRFs) if $\rho = 1 - \mathrm{o}(1)$. This argument is from [GG24, Section 3].

Let $\ell = (\log \lambda)/(1-\rho)$ and $\mathsf{PRF}_{\mathsf{sk}} : \{0,1\}^\ell \to \{0,1\}^\ell$ be a PRF. Our encoder $\mathsf{Enc}(\mathsf{sk})$ first samples random strings $r_1, \ldots, r_{\lambda^2} \leftarrow \{0,1\}^\ell$, then outputs

$$c = (r_1 \| \mathsf{PRF}_{\mathsf{sk}}(r_1) \| \cdots \| r_{\lambda^2} \| \mathsf{PRF}_{\mathsf{sk}}(r_{\lambda^2})) .$$

The decoder, upon receiving $x = (\widetilde{r}_1 \| \widetilde{y}_1 \| \cdots \| \widetilde{r}_{\lambda^2} \| \widetilde{y}_{\lambda^2})$, simply checks if $\widetilde{y}_i = \mathsf{PRF}_{\mathsf{sk}}(\widetilde{r}_i)$ for any i.

Pseudorandomness follows from the security of the PRF and the fact that $\ell = \omega(\log \lambda)$ if $\rho = 1 - \mathrm{o}(1)$. For soundness, suppose that x is independent of sk: Then for any i, the probability that $\widetilde{y}_i = \mathsf{PRF}_{\mathsf{sk}}(\widetilde{r}_i)$ is at most $2^{-\ell} + \mathsf{negl}(\lambda)$ by security of the PRF. If $\rho = 1 - o(1)$ then this is $\mathsf{negl}(\lambda)$, and by a union bound so is the probability that the decoder outputs 1.

For robustness to $\mathcal{N}_\rho$, suppose that $c \leftarrow \mathsf{Enc}(\mathsf{sk})$ and $\tilde{c} \leftarrow \mathcal{N}(c)$. Then $\mathsf{Dec}(\mathsf{sk}, \tilde{c}) = \perp$ only if every block $r_i \| y_i$ contains an error. Each bit is correct with probability $\frac{1+\rho}{2}$, so each block contains an error with probability $1 - \left(\frac{1+\rho}{2}\right)^{2\ell}$. The probability that every block contains an error is therefore

$$\left(1 - \left(\frac{1+\rho}{2}\right)^{2\ell}\right)^{\lambda^2} \leqslant \left(1 - \frac{\lambda^{\rho^2/2}}{\lambda^2}\right)^{\lambda^2} = \mathsf{negl}(\lambda)$$

after simplifying.

This construction works when $\rho = 1 - \mathrm{o}(1)$, but it fundamentally breaks down if $\rho = 1 - \Omega(1)$. The issue is that pseudorandomness requires our seeds r_i to each have length at least $\ell = \omega(\log n)$ in order to avoid collisions; whereas any completeness-soundness gap requires $\ell = \mathcal{O}(\log n)$, because a constant noise rate will produce an error on a string of length ℓ with probability $1 - \exp(-\ell)$.

We show that this threshold is not a result of an insufficiently clever scheme, but a fundamental barrier. That is, while we have just seen that it is easy to build a PRC for any $\mathrm{o}(1)$ error rate in the random oracle model, it is impossible to build a PRC for any $\Omega(1)$ error rate with black-box reductions to standard generic assumptions.

2.3 The proof strategy

The remainder of our overview is devoted to proving the impossibility. The bulk of the proof is showing that queries to the random oracle are not useful for constructing a PRC.

In order to show that a random oracle is not useful in a two-party (e.g., encoder and decoder) protocol against a third-party eavesdropper (e.g., pseudorandomness distinguisher), a standard methodology in the black-box separation literature is to argue that the eavesdropper can learn all the *intersection queries* [IR89, BM09]. The intersection queries are those made by both parties engaging in the protocol; if the eavesdropper knows all the intersection queries, then the participating parties cannot establish a shared secret.

However, this is not possible in our setting. Unlike every prior work we are aware of, we are in the *secret-key* setting, where the encoder and decoder share sk. The encoder and decoder can simply both query, say, sk, making sk an intersection query that the eavesdropper has no way of learning. In fact, the encoder and decoder can prevent the eavesdropper from learning any query at all by prepending sk to all of their queries. One can therefore think of our model as a *secret random oracle model*, where only the secret key holders have access to the oracle. Our approach will have to demonstrate that the noise channel makes it impossible for the encoder and decoder to put this secret random oracle to good use.

Consider an arbitrary PRC for $\mathcal{N}_\rho$, say $(\mathsf{Enc}^{\mathcal{R}}, \mathsf{Dec}^{\mathcal{R}})$, that uses a random oracle $\mathcal{R} : \{0,1\}^\lambda \to \{0,1\}$. Instead of defining an adversary, we will compile this PRC into a new one, say $(\widehat{\mathsf{Enc}}^{\mathcal{R}}, \widehat{\mathsf{Dec}}^{\mathcal{R}})$, where the decoder uses *one fewer query* to $\mathcal{R}$, at the cost of a somewhat longer secret key and a small loss in completeness.[4] By iterating this transformation, we can eliminate all queries from the decoder to the oracle—resulting in a PRC that is easily seen to be impossible: If the decoder makes no queries, then the encoder can simulate the oracle locally, and the oracle can be removed altogether. But any PRC is in particular a pseudorandom encryption scheme, which cannot have statistical security in the standard model.

So, how does our compiled scheme $(\widehat{\mathsf{Enc}}^{\mathcal{R}}, \widehat{\mathsf{Dec}}^{\mathcal{R}})$ work? Let f_{sk} be the function mapping the received string x to the first query $\mathsf{Dec}^{\mathcal{R}}(\mathsf{sk}, x)$ makes to $\mathcal{R}$. Suppose for simplicity that f_{sk} is deterministic. The basic idea of our compiled scheme is to transfer the oracle values on all queries q such that $\Pr_{x \leftarrow \{0,1\}^n}[f_{\mathsf{sk}}(x) = q] \geqslant \tau$ to the secret key, for some threshold $\tau = 1/\mathsf{poly}(\lambda)$. We call such queries "low entropy" and otherwise "high entropy." Since there can be at most $1/\tau$ low entropy queries, our new secret key $\widehat{\mathsf{sk}}$ is at most $\mathcal{O}(\lambda/\tau)$ bits longer than sk.

The compiled scheme will simulate $(\mathsf{Enc}^{\mathcal{R}}, \mathsf{Dec}^{\mathcal{R}})$, responding to oracle queries as follows:

- For low entropy queries, both $\widehat{\mathsf{Enc}}$ and $\widehat{\mathsf{Dec}}$ consult $\widehat{\mathsf{sk}}$.
- If the first query made by Dec is a high entropy query (that is, it is not in $\widehat{\mathsf{sk}}$), then $\widehat{\mathsf{Dec}}$ samples a fresh random response.
- For all other queries, both $\widehat{\mathsf{Enc}}$ and $\widehat{\mathsf{Dec}}$ respond according to $\mathcal{R}$.

Observe that the first oracle query made by Dec is removed in $\widehat{\mathsf{Dec}}$, so our new decoder makes one fewer query. Pseudorandomness and soundness for the compiled scheme are immediate, so we must only consider completeness. Recall the completeness experiment:

[4] In Section 4 we will remove all queries at once. We take the one-query-at-a-time approach in this overview because we find it conceptually somewhat simpler at this level of technicality.

1. Sample $\widehat{\mathsf{sk}} \leftarrow \{0,1\}^\lambda$ uniformly at random.
2. Generate a codeword $c \leftarrow \widehat{\mathsf{Enc}}^{\mathcal{R}}(\widehat{\mathsf{sk}})$.
3. Add noise to the codeword, $\tilde{c} \leftarrow \mathcal{N}_\rho(c)$.
4. Apply the decoder, $\widehat{\mathsf{Dec}}^{\mathcal{R}}(\widehat{\mathsf{sk}}, \tilde{c})$. If the result is $\perp$, the experiment fails; if it is 1 then the experiment succeeds.

The only case where $(\widehat{\mathsf{Enc}}^{\mathcal{R}}, \widehat{\mathsf{Dec}}^{\mathcal{R}})$ might fail to perfectly simulate $(\mathsf{Enc}^{\mathcal{R}}, \mathsf{Dec}^{\mathcal{R}})$ is when the first query made by Dec, say q, is a high entropy query: If it happens that $\widehat{\mathsf{Enc}}$ had also queried q in Step 2, then $\widehat{\mathsf{Enc}}$ and $\widehat{\mathsf{Dec}}$ will use different values for $\mathcal{R}(q)$—potentially ruining completeness. We argue in Section 2.4 that the error channel $\mathcal{N}_\rho$ prevents $\widehat{\mathsf{Enc}}$ and $\widehat{\mathsf{Dec}}$ from making the same high entropy query.

2.4 Bounding the error of the compiler

Recall the function f_{sk}, which maps the string $\tilde{c}$ received in Step 4 to the first oracle query $\widehat{\mathsf{Dec}}$ makes. For this overview we assume f_{sk} is deterministic to simplify notation and terminology. If we let S be the set of high entropy queries[5] made by $\widehat{\mathsf{Enc}}^{\mathcal{R}}(\widehat{\mathsf{sk}})$ in step 2, then our aim is to show that the following probability is $\mathsf{negl}(\lambda)$:

$$\Pr_{\substack{c \leftarrow \widehat{\mathsf{Enc}}^{\mathcal{R}}(\widehat{\mathsf{sk}}) \\ \tilde{c} \leftarrow \mathcal{N}_\rho(c)}} [f_{\mathsf{sk}}(\tilde{c}) \in S]. \tag{1}$$

Again, if $f_{\mathsf{sk}}(\tilde{c}) \notin S$ then our compiled scheme perfectly simulates the original one, so this will complete our proof.

The first simplifying observation is that, while c has an unknown distribution for a given $(\widehat{\mathsf{sk}}, \mathcal{R})$, the *marginal* distribution on just $(\widehat{\mathsf{sk}}, c)$ is actually uniform. The reason is simple: If $c \leftarrow \widehat{\mathsf{Enc}}^{\mathcal{R}}(\widehat{\mathsf{sk}})$ was non-uniform conditioned on $\widehat{\mathsf{sk}}$ alone, then the decoder could (inefficiently) distinguish codewords from random strings without using $\mathcal{R}$ at all! But, as we mentioned earlier, statistically "pseudorandom" encryption is not possible in the standard model (even with an inefficient decoder). For a formal proof with quantitative bounds on the distance to uniformity, see Lemma 1.

So we have the following equivalence with Equation (1), where the codeword is now sampled *uniformly at random* instead of using the encoder:

$$\Pr_{\substack{c \leftarrow \widehat{\mathsf{Enc}}^{\mathcal{R}}(\widehat{\mathsf{sk}}) \\ \tilde{c} \leftarrow \mathcal{N}_\rho(c)}} [f_{\mathsf{sk}}(\tilde{c}) \in S] = \Pr_{\substack{x \leftarrow \{0,1\}^n \\ \tilde{x} \leftarrow \mathcal{N}_\rho(x)}} [f_{\mathsf{sk}}(\tilde{x}) \in S].$$

The set S of high entropy queries made by the encoder clearly does not depend on the error introduced by $\mathcal{N}_\rho$. So by a union bound, if we let Q be an upper bound on the number of queries made by the decoder (so $|S| \leqslant Q$),

$$\Pr_{\substack{x \leftarrow \{0,1\}^n \\ \tilde{x} \leftarrow \mathcal{N}_\rho(x)}} [f_{\mathsf{sk}}(\tilde{x}) \in S] \leqslant Q \cdot \max_{f,g \in \mathcal{H}_\tau} \Pr_{\substack{x \leftarrow \{0,1\}^n \\ \tilde{x} \leftarrow \mathcal{N}_\rho(x)}} [f(\tilde{x}) = g(x)],$$

[5] Remember that we define high entropy queries with respect to the decoder (specifically f_{sk}). If the decoder, run on a random input, is unlikely to make q as its first query, then q is a "high entropy query" even if the encoder queries q with probability 1.

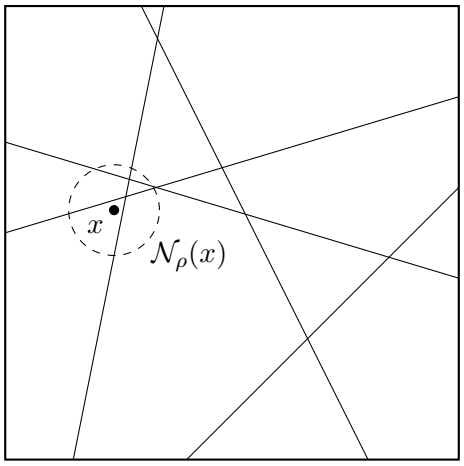

Figure 1: The space of strings $x \in \{0,1\}^n$ received by the decoder can be partitioned according to the first oracle query $f(x)$ the decoder makes upon receiving x. Equation (2) says that in high dimensions, if every cell in the partition is small, then a small Hamming ball centered at a random string has its mass distributed across many cells—i.e., no function $g(x)$ can guess where $\tilde{x} \leftarrow \mathcal{N}_\rho(x)$ will land. Thus the noise destroys the decoder's ability to make high-entropy intersection queries.

where $\mathcal{H}_\tau = \{f, g : \left|f^{-1}(g(x^*))\right|/2^n \leqslant \tau \text{ for all } x^* \in \{0,1\}^n\}$, capturing the fact that every $q \in S$ is at most τ-likely to be output by f.

Now it suffices to show the following simple and completely self-contained mathematical statement: For every $\rho = 1 - \Omega(1)$ and every pair of functions $f, g \in \mathcal{H}_\tau$,

$$\Pr_{\substack{x \leftarrow \{0,1\}^n \\ \tilde{x} \leftarrow \mathcal{N}_\rho(x)}}[f(\tilde{x}) = g(x)] = \tau^{\Omega(1)}. \tag{2}$$

Since τ is an arbitrary inverse-polynomial, this will imply that Equation (1) is $\mathsf{negl}(\lambda)$.

We will formally prove Equation (2) using hypercontractivity later, in Lemma 2. For now, let us just try to give some basic intuition, aided by Figure 1. What we need to show is that if two strings x and $\tilde{x}$ are each random on their own and only ρ-correlated together, then very little information can be consistently extracted from them.[6] To understand why this is true, consider the special case that $f = g$. We can view f as a partition of $\{0,1\}^n$, with a label on each cell corresponding to the output of f on any string in the cell. Since f is high entropy (i.e., $(f, f) \in \mathcal{H}_\tau$), each cell in the partition is small. The key point is that in the hypercube $\{0,1\}^n$, almost all of the probability mass is on the cell boundaries—therefore $f(x) \neq f(\tilde{x})$ with high probability. Proving that small subsets of the hypercube have mass concentrated on their boundaries is highly non-trivial, and is encapsulated in the hypercontractivity theorem.

2.5 Upgrading to a crypto oracle

It turns out that our random oracle impossibility easily extends to our full result by using the ideas of [LMW25]. That work introduces *crypto oracles* and shows that they are capable of implementing

[6] Note that it is essential that x and $\tilde{x}$ are each random on their own, as otherwise error correction would make it possible to consistently extract lots of information!

most generic cryptography. A crypto oracle is an efficient, stateless algorithm with access to a secret random oracle.

Because a PRC is a "secret-key" primitive—that is, the pseudorandomness adversary cannot access some secret randomness—our encoder and decoder can simply use the (public) random oracle to simulate the crypto oracle. That is, by prepending all of their queries to the random oracle with the secret key, the encoder and decoder can effectively share a secret random oracle.

Now any construction of a PRC relative to a crypto oracle $\mathcal{B}^{\mathcal{R}}$ immediately implies a PRC in the secret random oracle model, where one simply lets the encoder and decoder simulate $\mathcal{B}^{\mathcal{R}}$ using access to the secret oracle $\mathcal{R}$.[7] Consequently, our theorem immediately implies our separation result in the crypto oracle model.

3 Toolkit

3.1 Notation

We use λ for the security parameter and $\mathsf{negl}(\lambda)$ for a negligible function, i.e., for any polynomial $f(\lambda)$, it holds that $\mathsf{negl}(\lambda) < 1/f(\lambda)$ for all large enough λ. We use $\mathbb{N}$ to denote the set of positive integers. In this work "log" will denote the base-2 logarithm. We assume that the reader is familiar with the basic ideas of information theory, such as Shannon entropy, statistical distance, and KL divergence. For random variables X and Y, we write $H(X)$ for the binary entropy of X; $H(X|Y)$ for the binary entropy of X conditioned on Y; $\mathsf{SD}(X,Y)$ for the statistical distance (i.e. total variation distance) between X and Y; and $\mathrm{D}_{\mathrm{KL}}(X\|Y)$ for the binary KL divergence of X from Y.

For sets $\mathcal{X}, \mathcal{Y}$, we will write $f : \mathcal{X} \to \mathcal{Y}$ and say that f is a "randomized function" if, for each $x \in \mathcal{X}$, $f(x)$ is a random variable on $\mathcal{Y}$. We use this terminology for convenience of notation; we find that the more standard approach of having f map $\mathcal{X}$ to distributions on $\mathcal{Y}$ would be cumbersome, and calling f a "randomized algorithm" is not appropriate in all of our instances.

3.2 Information theory

In this section we prove Lemma 1, which we will use twice in our main proof: Once in order to invoke our key technical lemma, Lemma 2; and once in order to see that statistical security is impossible in the plain model. First we relate the Shannon entropy to the statistical distance from the uniform distribution.

Fact 1. *Let X be any random variable taking values in a finite set $\mathcal{S}$, and let R be a uniformly random sample from $\mathcal{S}$. If $\varepsilon = \mathsf{SD}(X,R) \leqslant 1/4$, then*

$$2\varepsilon^2 \leqslant \log|S| - H(X) \leqslant 2\varepsilon \log|S| + 2\sqrt{\varepsilon}.$$

Proof. For the inequality on the left, first observe that $\mathrm{D}_{\mathrm{KL}}(X\|R) = \log|S| - H(X)$. Then by Pinsker's inequality, $2\varepsilon^2 \leqslant \mathrm{D}_{\mathrm{KL}}(X\|R) \cdot \ln 2 \leqslant \mathrm{D}_{\mathrm{KL}}(X\|R)$.

For the inequality on the right, we have $\log|S| - H(X) \leqslant 2\varepsilon \log|S| + 2\varepsilon \log(1/2\varepsilon)$ from [CT01, Theorem 16.3.2]. We simplify the second term as $2\varepsilon \log(1/2\varepsilon) \leqslant 2\varepsilon \log(1/\varepsilon) \leqslant 2\sqrt{\varepsilon}$ for $\varepsilon \leqslant 1/4$. □

[7] In a crypto oracle model, a PRC guarantees that an adversary cannot break pseudorandomness given access to $\mathcal{B}^{\mathcal{R}}$. In the compiled scheme, the adversary does not even have access to $\mathcal{B}^{\mathcal{R}}$ and, hence, can only become weaker and achieve a lower advantage.

We now deduce Lemma 1. Roughly, Lemma 1 states that a finite-length key can only be used to statistically hide a finite number of messages.

Lemma 1. *Let $\{\mathcal{D}_{\mathsf{k}}\}_{\mathsf{k}\in\{0,1\}^\ell}$ be a family of distributions over $\{0,1\}^n$. Let sk be a uniformly random sample from $\{0,1\}^\ell$ and let $x, x_1, \ldots, x_m$ be independent samples from $\mathcal{D}_{\mathsf{sk}}$.*

If $(x_1, \ldots, x_m)$ are jointly ε-close to uniform in statistical distance, then (sk, x) is

$$\mathcal{O}\left(\sqrt{\varepsilon n + \sqrt{\varepsilon}/m + \ell/m}\right)$$

close to uniform in statistical distance.

Proof. Let sk be a random string from $\{0,1\}^\ell$, and let $x, x_1, \ldots, x_m$ be samples from $\mathcal{D}_{\mathsf{sk}}$. We have

$$\begin{aligned} H(x_1, \ldots, x_m) &\leqslant H(x_1, \ldots, x_m, \mathsf{sk}) \\ &= H(\mathsf{sk}) + H(x_1, \ldots, x_m|\mathsf{sk}) \\ &= \ell + m \cdot H(x|\mathsf{sk}), \end{aligned}$$

which means that $H(x|\mathsf{sk}) \geqslant (H(x_1, \ldots, x_m) - \ell)/m$.

Let R be a uniformly random sample from $\{0,1\}^{\ell+n}$. Fact 1 implies the following two inequalities:

- $\mathsf{SD}\left((\mathsf{sk}, x), R\right) \leqslant \sqrt{n + \ell - H(\mathsf{sk}, x)} = \sqrt{n - H(x|\mathsf{sk})}$, and
- $H(x_1, \ldots, x_m) \geqslant nm - 2\varepsilon nm - 2\sqrt{\varepsilon}$.

Therefore

$$\begin{aligned} \mathsf{SD}\left((\mathsf{sk}, x), R\right) &\leqslant \sqrt{n - H(x|\mathsf{sk})} \\ &\leqslant \sqrt{n - (H(x_1, \ldots, x_m) - \ell)/m} \\ &\leqslant \sqrt{n - (nm - 2\varepsilon nm - 2\sqrt{\varepsilon} - \ell)/m} \\ &= \sqrt{2\varepsilon n + 2\sqrt{\varepsilon}/m + \ell/m}, \end{aligned}$$

completing the proof. □

3.3 Pseudorandom codes

Following [CG24], we define private-key pseudorandom codes as follows. Our impossibility result will apply to the private-key setting, which immediately implies the corresponding impossibility for the public-key setting. Therefore we omit definitions of public-key pseudorandom codes.

Definition 1 (Private-key PRC)**.** *A* private-key $(\delta, \varepsilon, \mu)$ pseudorandom error-correcting code *over an alphabet Σ and a channel $\mathcal{E}\colon \Sigma^* \to \Sigma^*$ consists of a tuple of PPT algorithms* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*:*

- *A key generation algorithm* KeyGen *that samples a secret key* $\mathsf{sk} \in \{0,1\}^{\ell(\lambda)}$.
- *An encoding algorithm* Enc *that takes the secret key* $\mathsf{sk} \in \{0,1\}^{\ell(\lambda)}$ *and the message* $m \in \Sigma^{k(\lambda)}$ *as input and outputs a codeword* $c \in \Sigma^{n(\lambda)}$.

- *A decoding algorithm that takes the secret key* $\mathsf{sk} \in \{0,1\}^{\ell(\lambda)}$ *and a (potentially erroneous) codeword* $c \in \Sigma^{n(\lambda)}$ *as input and outputs a message* $m \in \Sigma^{k(\lambda)} \cup \{\perp\}$.

These algorithms must satisfy the following properties:

- δ***-Completeness (robustness).*** *For any message* $m \in \Sigma^k$*, it holds that*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, \widetilde{c}) = m \;\middle|\; \begin{array}{c} \mathsf{sk} \leftarrow \mathsf{KeyGen}(\cdot) \\ c \leftarrow \mathsf{Enc}(\mathsf{sk}, m) \\ \widetilde{c} \leftarrow \mathcal{E}(c) \end{array}\right] \geqslant 1 - \delta.$$

- ε***-Pseudorandomness.*** *For any PPT adversary* $\mathcal{A}$*, it holds that*

$$\left|\Pr\left[\mathcal{A}^{\mathsf{Enc}(\mathsf{sk},\cdot)}(\cdot) = 1 \;\middle|\; \mathsf{sk} \leftarrow \mathsf{KeyGen}(\cdot)\right] - \Pr\left[\mathcal{A}^{\mathcal{U}(\cdot)}(\cdot) = 1\right]\right| \leqslant \varepsilon,$$

  where $\mathsf{Enc}(\mathsf{sk}, \cdot)$ *generates fresh encodings even if the same message is queried twice and* $\mathcal{U}(\cdot)$ *is an oracle that simply outputs fresh random strings.*

- μ***-Soundness.*** *For a fixed codeword* $c^* \in \Sigma^n$*, it holds that*

$$\Pr[\mathsf{Dec}(\mathsf{sk}, c^*) = \perp \mid \mathsf{sk} \leftarrow \mathsf{KeyGen}(\cdot)] \geqslant 1 - \mu.$$

We will drop the specification of δ, ε, μ *in the case that they are all bounded by* $\mathsf{negl}(\lambda)$.

Zero-bit PRC. We consider PRCs with only one possible message, $m = 1$, which are known as *zero-bit* PRCs and are useful for watermarking. For random errors, one can convert a zero-bit PRC to a many-bit PRC simply by concatenating codewords and random strings [CG24]. Conversely, a zero-bit PRC is immediately implied by any PRC with $k > 0$, so our impossibility result is not weakened at all by restricting to this case. We therefore drop the message in the remainder of this paper, writing $\mathsf{Enc}(\mathsf{sk})$ instead of $\mathsf{Enc}(\mathsf{sk}, 1)$. The decoder $\mathsf{Dec}(\mathsf{sk}, x)$ outputs 1 or $\perp$, indicating success or failure, respectively.

Binary Alphabet. In this work we will restrict ourselves to a binary alphabet, i.e. $\Sigma = \{0,1\}$. Note that a pseudorandom code over large alphabet implies a pseudorandom code for binary alphabet. Since we are proving an impossibility result, this restriction only makes our lower bound stronger.

Noise channel. In this work we will only consider the binary symmetric channel, which introduces i.i.d bit-flip errors on the codeword bits. Adversarial noise channels such as those considered in [AAC+25] can easily simulate the binary symmetric channel, so our impossibility result applies to pseudorandom codes for those channels as well. We will denote the binary symmetric channel by $\mathcal{N}_\rho$, defined as

$$\mathcal{N}_\rho(x) = x \oplus e$$

where $e \leftarrow \mathsf{Ber}((1-\rho)/2)^n$. We use $\mathsf{Ber}(p)$ to denote the Bernoulli distribution with probability p, and $\oplus$ to denote the bitwise XOR. In other words, each symbol of x is replaced by a random element with probability $1 - \rho$ in $\mathcal{N}_\rho(x)$.

3.4 Boolean analysis and hypercontractivity

A core part of our argument makes use of Boolean analysis. We introduce minimal notation here to facilitate our proof; we refer the reader to [O'D14] for more details. For any Boolean function $f : \{0,1\}^n \to \mathbb{R}$, its ℓ-norm $\|f\|_\ell$ is defined as

$$\|f\|_\ell = \left(\mathop{\mathbb{E}}_x[f(x)^\ell]\right)^{1/\ell}.$$

Unless otherwise specified, expectations are assumed to be taken over the uniform distribution. For any two Boolean functions f and g, their *inner product* is defined as

$$\langle f, g\rangle = \mathop{\mathbb{E}}_x[f(x) \cdot g(x)].$$

By Cauchy-Schwartz, we have

$$\langle f, g\rangle \leqslant \|f\|_2 \cdot \|g\|_2.$$

We now define the *noise operator* T_ρ. For any Boolean function f, the noise operator T_ρ on the function f defines a new function $T_\rho f$ as

$$(T_\rho f)(x) = \mathop{\mathbb{E}}_{y \leftarrow \mathcal{N}_\rho(x)}[f(y)].$$

The following is a special case of the *hypercontractivity theorem* [KKL88].

Theorem 1 (Hypercontractivity, [KKL88])**.** *For any function $f : \{0,1\}^n \to \mathbb{R}$ and $\rho \in [0,1]$,*

$$\|T_\rho f\|_2 \leqslant \|f\|_{1+\rho^2}.$$

Hypercontractivity is the core ingredient in our main technical lemma, Lemma 2.

Lemma 2. *Let $\alpha > 0$. Suppose $f, g : \{0,1\}^n \to \mathcal{Y}$ are (randomized) functions such that, for all $x^* \in \{0,1\}^n$,*

$$\Pr[f(x) = g(x^*) \mid x \leftarrow \{0,1\}^n] \leqslant \alpha.$$

Then, for any $\rho \in [0,1]$, it holds that

$$\Pr\left[f(\widetilde{x}) = g(x) \;\middle|\; \begin{array}{c} x \leftarrow \{0,1\}^n \\ \widetilde{x} \leftarrow \mathcal{N}_\rho(x) \end{array}\right] \leqslant \alpha^{\frac{1}{2}\cdot\left(\frac{1-\rho^2}{1+\rho^2}\right)}.$$

Proof. The proof works by a careful application of Cauchy-Schwarz and hypercontractivity. For $y \in \mathcal{Y}$ and $x \in \{0,1\}^n$, let $p_y(x) = \Pr[f(x) = y]$ and $q_y(x) = \Pr[g(x) = y]$. Let $\mathcal{Y}^* = \{y \in \mathcal{Y} :$

$\|q_y\|_1 > 0\}$.

$$
\begin{aligned}
&\Pr\left[f(\widetilde{x}) = g(x) \;\middle|\; \begin{array}{l} x \leftarrow \{0,1\}^n \\ \widetilde{x} \leftarrow \mathcal{N}_\rho(x) \end{array} \right] \\
&= \mathop{\mathbb{E}}_{x \leftarrow \{0,1\}^n} \sum_{y \in \mathcal{Y}^*} q_y(x) \cdot (T_\rho p_y)(x) \\
&= \sum_{y \in \mathcal{Y}^*} \langle q_y, T_\rho p_y \rangle \\
&\leqslant \sum_{y \in \mathcal{Y}^*} \|q_y\|_2 \cdot \|T_\rho p_y\|_2 && \text{(Cauchy-Schwarz)} \\
&\leqslant \sum_{y \in \mathcal{Y}^*} \|q_y\|_2 \cdot \|p_y\|_{1+\rho^2} && \text{(Hypercontractivity)} \\
&\leqslant \sqrt{\sum_{y \in \mathcal{Y}^*} \|q_y\|_2^2} \cdot \sqrt{\sum_{y \in \mathcal{Y}^*} \|p_y\|_{1+\rho^2}^2} && \text{(Cauchy-Schwarz)} \\
&= \sqrt{\sum_{y \in \mathcal{Y}^*} \mathop{\mathbb{E}}_x q_y(x)^2} \cdot \sqrt{\sum_{y \in \mathcal{Y}^*} \left(\mathop{\mathbb{E}}_x p_y(x)^{1+\rho^2} \right)^{2/(1+\rho^2)}} \\
&\leqslant \sqrt{\sum_{y \in \mathcal{Y}^*} \mathop{\mathbb{E}}_x q_y(x)} \cdot \sqrt{\sum_{y \in \mathcal{Y}^*} \left(\mathop{\mathbb{E}}_x p_y(x) \right)^{2/(1+\rho^2)}}
\end{aligned}
$$

Letting $P(y) = \Pr_{x \leftarrow \{0,1\}^n}[f(x) = y]$ and $Q(y) = \Pr_{x \leftarrow \{0,1\}^n}[g(x) = y]$,

$$
\begin{aligned}
&= \sqrt{\sum_{y \in \mathcal{Y}^*} Q(y)} \cdot \sqrt{\sum_{y \in \mathcal{Y}^*} P(y)^{2/(1+\rho^2)}} \\
&= 1 \cdot \sqrt{\sum_{y \in \mathcal{Y}^*} P(y) \cdot P(y)^{2/(1+\rho^2)-1}} \\
&\leqslant \sqrt{\sum_{y \in \mathcal{Y}^*} P(y) \cdot \alpha^{2/(1+\rho^2)-1}} && \text{(Assumption on } f\text{)} \\
&= \alpha^{\frac{1}{2}\left(\frac{1-\rho^2}{1+\rho^2}\right)}. && \square
\end{aligned}
$$

3.5 Crypto oracles

The recent work of [LMW25] introduced "crypto oracles" for the purposes of an impossibility result for doubly-efficient private information retrieval.

Definition 2 (Crypto oracle, [LMW25])**.** *A* crypto oracle *is a function $\mathcal{B}^{\mathcal{R}}$ where $\mathcal{B}$ is a stateless, polynomial time, deterministic Turing machine with oracle access to a secret random function $\mathcal{R} : \{0,1\}^* \to \{0,1\}$.*

The same work proved the following, which we have heavily paraphrased for simplicity. See their paper for details.

Theorem 2 ([LMW25]). *There exist crypto oracles that implement:*

- *VBB obfuscation for Turing machines, and*
- *the generic multilinear group.*

4 The impossibility

4.1 Compiling out the random oracle

In this section, we present a compiler that compiles any PRC in the random oracle model into a PRC in the information-theoretic setting. The new scheme has the same soundness and pseudorandomness, but it requires a larger secret key and incurs a loss in the completeness error.

We assume $\mathsf{KeyGen}^{\mathcal{R}}$ does not make any oracle queries. This is without loss of generality because we can simply include any $\mathcal{R}$ queries/responses used by KeyGen in the secret key.

In the following lemma and proof, we leave the dependence of functions of the security parameter on the security parameter implicit, writing e.g. τ instead of $\tau(\lambda)$.

Lemma 3. *Let $\ell, Q : \mathbb{N} \to \mathbb{N}$ be functions of the security parameter. Let $(\mathsf{KeyGen}, \mathsf{Enc}^{\mathcal{R}}, \mathsf{Dec}^{\mathcal{R}})$ be any zero-bit $(\delta, \varepsilon, \mu)$-PRC construction in the random oracle model where $\mathsf{Enc}^{\mathcal{R}}$ and $\mathsf{Dec}^{\mathcal{R}}$ each make at most Q queries to $\mathcal{R}$ and $\mathsf{KeyGen}(1^\lambda)$ outputs keys of length at most ℓ.*

Then for any $\tau : \mathbb{N} \to (0,1)$, there exists a $(\delta', \varepsilon', \mu')$-PRC construction $(\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}')$ in the standard model where

- $\delta' = \delta + 2^{-\lambda} \cdot Q/\tau + Q^2 \cdot \tau^{\frac{1}{2}\cdot\left(\frac{1-\rho^2}{1+\rho^2}\right)} + \mathcal{O}\big(Q^2 \cdot \sqrt{\varepsilon n}\big)$,
- $\varepsilon' = \varepsilon$,
- $\mu' = \mu$, *and*
- *the size of the new secret key is $\ell + \mathcal{O}\big(Q \cdot \lambda^2/\tau\big)$.*

Proof of Lemma 3. Our compiler is presented in Figure 2. We now proceed to prove its properties as stated in Lemma 3.

Secret key size. The size of the secret key grows by the size of S, which is $\mathcal{O}\big(Q \cdot \lambda^2/\tau\big)$ by construction.

Pseudorandomness & soundness. Observe that for the pseudorandomness (resp., soundness) property, the experiment only involves KeyGen' and the encoder Enc' (resp., the decoder Dec'). The distribution of any process involving only the encoder (resp., decoder) is identical in the compiled scheme to the original scheme. This is because *locally*, the encoder (resp., decoder) perfectly simulates the random oracle. Consequently, the pseudorandomness and soundness guarantees do not change at all.

Completeness. If Enc^{F_1} and Dec^{F_2} do not make the same query (except for those contained in S), then the compiled scheme perfectly simulates the original scheme. Therefore, it suffices to

Let $\tau \in (0,1)$ be any number. Let $(\mathsf{KeyGen}, \mathsf{Enc}^{\mathcal{R}}, \mathsf{Dec}^{\mathcal{R}})$ be a PRC in the random oracle model where $\mathsf{Dec}^{\mathcal{R}}$ makes at most $Q = Q(\lambda)$ queries to $\mathcal{R}$. We compile it into a new PRC scheme $(\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}')$ in the standard model as follows.

- $\mathsf{KeyGen}'(1^\lambda)$:
  1. Sample $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda)$ and a random function $F_0 : \{0,1\}^\lambda \to \{0,1\}$.
  2. Initialize $S = \emptyset$ and repeat the following for $\lceil \lambda/\tau \rceil$ times:
     - Sample $x \leftarrow \{0,1\}^n$ and run $\mathsf{Dec}^{F_0}(\mathsf{sk}, x)$, recording each query-response pair in S.
  3. Output the new secret key as $\mathsf{sk}' = (\mathsf{sk}, S)$.
- $\mathsf{Enc}'(\mathsf{sk}')$: Parse $\mathsf{sk}' = (\mathsf{sk}, S)$ and sample a random function $F_1 : \{0,1\}^\lambda \to \{0,1\}$ that is consistent with S. Run $\mathsf{Enc}^{F_1}(\mathsf{sk})$ and output the result.
- $\mathsf{Dec}'(\mathsf{sk}', x)$: Parse $\mathsf{sk}' = (\mathsf{sk}, S)$ and sample a random function $F_2 : \{0,1\}^\lambda \to \{0,1\}$ that is consistent with S. Run $\mathsf{Dec}^{F_2}(\mathsf{sk}, x)$ and output the result.

Figure 2: Our compiler

bound the probability that Enc^{F_1} and Dec^{F_2} both query their oracles on the same point q which is not in S. We refer to such a query as an *intersection query.*[8]

We will say that a query q is τ-*heavy* for sk, F if

$$\Pr\left[\mathsf{Dec}^{F}(\mathsf{sk}, x) \text{ queries } q \mid x \leftarrow \{0,1\}^n\right] \geqslant \tau.$$

Consider the PRC completeness experiment for our compiled scheme.

Completeness experiment:

1. Sample $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda)$ and a random function $F_0 : \{0,1\}^\lambda \to \{0,1\}$.
2. Initialize $S = \emptyset$ and repeat the following for $\lceil \lambda/\tau \rceil$ times:
   - Sample $x \leftarrow \{0,1\}^n$ and run $\mathsf{Dec}^{F_0}(\mathsf{sk}, x)$. Record each query-response pair in S.
3. Let $\mathsf{sk}' = (\mathsf{sk}, S)$. Sample random functions $F_1, F_2 : \{0,1\}^\lambda \to \{0,1\}$ consistent with S.
4. Compute $c \leftarrow \mathsf{Enc}^{F_1}(\mathsf{sk})$ and $\tilde{c} \leftarrow \mathcal{N}_\rho(c)$.
5. Compute $\mathsf{Dec}^{F_2}(\mathsf{sk}, \tilde{c})$.

In this experiment we define two events:

- Bad_1: This event happens when S does not contain all queries which are τ-heavy for sk, F_2.

[8] Note that queries in S are not counted as intersection queries.

- Bad_2: This event happens when (1) S contains all queries which are τ-heavy for sk, F_2, but (2) $\mathsf{Enc}^{F_1}(\mathsf{sk})$ and $\mathsf{Dec}^{F_2}(\mathsf{sk}, \tilde{c})$ still make an intersection query (which is not included in S).

We conclude the completeness guarantee by proving the following two bounds:

1. $\Pr[\mathsf{Bad}_1] \leqslant 2^{-\lambda} \cdot Q/\tau$. This bound will hold for every fixed choice of sk.
2. $\Pr[\mathsf{Bad}_2] \leqslant Q^2 \cdot \left(\tau^{\frac{1}{2} \cdot \left(\frac{1-\rho^2}{1+\rho^2}\right)} + \mathcal{O}(\sqrt{\varepsilon n}) \right)$. This bound will only hold on average over sk.

Bounding Bad_1. In this part we will consider sk as fixed. For any $q \in \{0,1\}^\lambda$ that is τ-heavy for sk, F_0, the probability that q is queried in any single iteration of the loop in step 2 is at least τ. Therefore, the probability that q is *never* queried throughout the $\lceil \lambda/\tau \rceil$ iterations of the loop in step 2 is at most

$$(1-\tau)^{\lceil \lambda/\tau \rceil} \leqslant 2^{-\lambda}.$$

The total number of τ-heavy queries for sk, F_0 is upper bounded by Q/τ because $\mathsf{Dec}^{F_0}(\mathsf{sk}, \cdot)$ makes at most Q queries. So by a union bound, the probability that S does not contain all the queries which are τ-heavy for sk, F_0 is at most $2^{-\lambda} \cdot Q/\tau$.

Of course, we are interested in the probability that S contains all the queries which are τ-heavy for sk, F_2 (not sk, F_0). But these two probabilities are the same because, conditioned on S, F_0 and F_2 are equal on every point in S and both uniformly random on every other point.

More formally, note that for a given sk we could equivalently define F_0 as follows:

1. Initialize $S = \emptyset$ and repeat the following for $\lceil \lambda/\tau \rceil$ times:
   - Sample $x \leftarrow \{0,1\}^n$ and run $\mathsf{Dec}^{(\cdot)}(\mathsf{sk}, x)$, simulating the oracle on-the-fly. Record each query-response pair in S.
2. Let $\mathsf{sk}' = (\mathsf{sk}, S)$. Sample a random function $F_0 : \{0,1\}^\lambda \to \{0,1\}$ consistent with S.

Since the distribution of S, F_0 does not change if we sample F_0 in this way instead, the probability that S contains all the queries which are τ-heavy for sk, F_0 does not change. However, under this method of sampling it is clear that S, F_0 is distributed identically to S, F_2, so

$$\Pr[\mathsf{Bad}_1] \leqslant 2^{-\lambda} \cdot Q/\tau.$$

Bounding Bad_2. Now we are interested in the probability that both the encoder and decoder ask a query that is not τ-heavy for sk, F_2. Whereas we bounded $\Pr[\mathsf{Bad}_1]$ for every fixed choice of sk, we will use the randomness of the entire experiment to bound $\Pr[\mathsf{Bad}_2]$.

For $i, j \in [Q]$, let $\mathsf{Bad}_2^{i,j}$ denote the event that (1) S contains all the τ-heavy queries for sk, F_2, but (2) the i-th query made by $\mathsf{Enc}^{F_1}(\mathsf{sk})$ and the j-th query made by $\mathsf{Dec}^{F_2}(\mathsf{sk}, \tilde{c})$ are both some q that is not in S. By a union bound,

$$\Pr[\mathsf{Bad}_2] \leqslant \sum_{i,j \in [Q]} \Pr\left[\mathsf{Bad}_2^{i,j}\right].$$

We will use Lemma 2 to show that, for every $i, j \in [Q]$,

$$\Pr\left[\mathsf{Bad}_2^{i,j}\right] \leqslant \tau^{\frac{1}{2} \cdot \left(\frac{1-\rho^2}{1+\rho^2}\right)} + \mathcal{O}\left(\sqrt{\varepsilon n}\right).$$

Let $f_{\mathsf{sk}'}$ be the (randomized) function with $\mathsf{sk}' = (S, \mathsf{sk})$ hardwired that maps $\tilde{c}$ to the j-th query made by $\mathsf{Dec}^{F_2}(\mathsf{sk}, \tilde{c})$. Let $h_{\mathsf{sk}'}$ be the (randomized) function with sk' hardwired that maps c to the i-th query made by $\mathsf{Enc}^{F_1}(\mathsf{sk})$, conditioned on $c \leftarrow \mathsf{Enc}^{F_1}(\mathsf{sk})$; let $g_{\mathsf{sk}'}$ be identical to $h_{\mathsf{sk}'}$ except that $g_{\mathsf{sk}'}(c)$ outputs $\perp$ whenever $h_{\mathsf{sk}'}$ outputs any query that is in S. Observe that

$$\Pr\left[\mathsf{Bad}_2^{i,j}\right] = \Pr\left[\neg\mathsf{Bad}_1 \wedge f_{\mathsf{sk}'}(\tilde{c}) = g_{\mathsf{sk}'}(c) \;\middle|\; \begin{array}{c} c \leftarrow \mathsf{Enc}^{F_1}(\mathsf{sk}) \\ \tilde{c} \leftarrow \mathcal{N}_\rho(c) \end{array}\right].$$

Applying Lemma 1, we have that $\mathsf{SD}\left((\mathsf{sk}', c \leftarrow \mathsf{Enc}(\mathsf{sk}')), (\mathsf{sk}', x \leftarrow \{0,1\}^n)\right) = \mathcal{O}(\sqrt{\varepsilon n})$, so

$$\begin{aligned}
&\Pr\left[\neg\mathsf{Bad}_1 \wedge f_{\mathsf{sk}'}(\tilde{c}) = g_{\mathsf{sk}'}(c) \;\middle|\; \begin{array}{c} c \leftarrow \mathsf{Enc}^{F_1}(\mathsf{sk}) \\ \tilde{c} \leftarrow \mathcal{N}_\rho(c) \end{array}\right] \\
&\leqslant \Pr\left[\neg\mathsf{Bad}_1 \wedge f_{\mathsf{sk}'}(\widetilde{x}) = g_{\mathsf{sk}'}(x) \;\middle|\; \begin{array}{c} x \leftarrow \{0,1\}^n \\ \widetilde{x} \leftarrow \mathcal{N}_\rho(x) \end{array}\right] + \mathcal{O}\left(\sqrt{\varepsilon n}\right).
\end{aligned}$$

Recall that $\neg\mathsf{Bad}_1$ means S contains all the τ-heavy queries for S, F_2, and $g_{\mathsf{sk}'}(x)$ never outputs any query that is in S. Therefore, for any $x^* \in \{0,1\}^n$,

$$\Pr\left[f_{\mathsf{sk}'}(x) = g_{\mathsf{sk}'}(x^*) \mid x \leftarrow \{0,1\}^n \wedge \neg\mathsf{Bad}_1\right] \leqslant \tau.$$

Since conditioning on $\neg\mathsf{Bad}_1$ does not affect the distribution on $x \leftarrow \{0,1\}^n, \tilde{x} \leftarrow \mathcal{N}_\rho(x)$, Lemma 2 implies that

$$\begin{aligned}
&\Pr\left[\neg\mathsf{Bad}_1 \wedge f_{\mathsf{sk}'}(\tilde{x}) = g_{\mathsf{sk}'}(x) \;\middle|\; \begin{array}{c} x \leftarrow \{0,1\}^n \\ \tilde{x} \leftarrow \mathcal{N}_\rho(x) \end{array}\right] \\
&\leqslant \Pr\left[f_{\mathsf{sk}'}(\tilde{x}) = g_{\mathsf{sk}'}(x) \;\middle|\; \begin{array}{c} x \leftarrow \{0,1\}^n \\ \tilde{x} \leftarrow \mathcal{N}_\rho(x) \\ \neg\mathsf{Bad}_1 \end{array}\right] \\
&\leqslant \tau^{\frac{1}{2}\cdot\left(\frac{1-\rho^2}{1+\rho^2}\right)},
\end{aligned}$$

completing the proof. □

4.2 The main theorem

Let $\mathcal{R} : \{0,1\}^\lambda \to \{0,1\}$ be a random oracle which is only given to the algorithms of the pseudorandom code. That is, we do not assume that pseudorandomness holds against adversaries who are allowed to access $\mathcal{R}$. We are now prepared to prove the following theorem.

Theorem 3. *Let* $(\mathsf{KeyGen}^{\mathcal{R}}, \mathsf{Enc}^{\mathcal{R}}, \mathsf{Dec}^{\mathcal{R}})$ *be a zero-bit* $(\delta, \varepsilon, \mu)$ *pseudorandom error-correcting code making queries to a secret random oracle* $\mathcal{R}$*. Suppose that the code is robust to the* ρ*-noise channel, and that the number of queries made by one execution of* $\mathsf{KeyGen}^{\mathcal{R}}$*,* $\mathsf{Enc}^{\mathcal{R}}$*, and* $\mathsf{Dec}^{\mathcal{R}}$ *is at most* Q*. Then*

$$\delta + \mu \geqslant 1 - Q^2 \cdot \lambda^{-\omega(1-\rho)} - \mathcal{O}\left(Q^2\sqrt{\varepsilon n}\right).$$

Proof. We first remove any oracle queries made by $\mathsf{KeyGen}^{\mathcal{R}}$ by sampling the responses at random and including them in the secret key. If $\mathsf{Enc}^{\mathcal{R}}$ or $\mathsf{Dec}^{\mathcal{R}}$ tries to make any query whose response is included in the secret key, it uses the stored response instead of consulting $\mathcal{R}$. This transformation does not change the parameters of our PRC at all, except to increase the length of the secret key by a polynomial in λ.

Next, we apply Lemma 3 using $\tau = \lambda^{-c}$, where $c > 0$ is any constant. This yields a statistically-pseudorandom $(\delta', \varepsilon, \mu)$-PRC construction $(\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}')$ in the standard model with

$$\delta' = \delta + \mathsf{negl}(\lambda) + Q^2 \cdot \lambda^{-\frac{c}{2} \cdot \left(\frac{1-\rho^2}{1+\rho^2}\right)} + \mathcal{O}\big(Q^2 \cdot \sqrt{\varepsilon n}\big)$$

and $\mathsf{poly}(\lambda)$-length secret keys. Now by Lemma 1, PRC codewords are $\mathcal{O}(\sqrt{\varepsilon n})$-indistinguishable from random strings *even to a distinguisher provided with the secret key.* Therefore the completeness-soundness gap is $\mathcal{O}(\sqrt{\varepsilon n})$, i.e., $\delta' + \mu \geqslant 1 - \mathcal{O}(\sqrt{\varepsilon n})$. Substituting the above formula for δ' and simplifying yields the result. □

Corollary 4. *There do not exist statistically-secure pseudorandom codes for any constant-rate noise channel, relative to any crypto oracle.*

Proof. Suppose that $(\mathsf{KeyGen}^{\mathcal{O}}, \mathsf{Enc}^{\mathcal{O}}, \mathsf{Dec}^{\mathcal{O}})$ is a pseudorandom code for some $(1 - \Omega(1))$-noise channel, relative to some crypto oracle $\mathcal{O}$. Assume it is a zero-bit PRC, by always using the all-zero message and disregarding the decoder's output.

Define $(\overline{\mathsf{KeyGen}}^{\mathcal{R}}, \overline{\mathsf{Enc}}^{\mathcal{R}}, \overline{\mathsf{Dec}}^{\mathcal{R}})$ to behave identically to $(\mathsf{KeyGen}^{\mathcal{O}}, \mathsf{Enc}^{\mathcal{O}}, \mathsf{Dec}^{\mathcal{O}})$, except that they simulate $\mathcal{O}$ using oracle access to a random function $\mathcal{R}$. If $\mathcal{R}$ is secret—that is, the pseudorandomness adversary is not allowed to query $\mathcal{R}$—then $(\overline{\mathsf{KeyGen}}^{\mathcal{R}}, \overline{\mathsf{Enc}}^{\mathcal{R}}, \overline{\mathsf{Dec}}^{\mathcal{R}})$ is a pseudorandom code. We can therefore invoke Theorem 3 with $(\overline{\mathsf{KeyGen}}^{\mathcal{R}}, \overline{\mathsf{Enc}}^{\mathcal{R}}, \overline{\mathsf{Dec}}^{\mathcal{R}})$ to conclude that

$$\delta + \mu \geqslant 1 - \mathsf{negl}(\lambda),$$

which means that there is a negligible completeness-soundness gap. □

References

[AAC+25] Omar Alrabiah, Prabhanjan Ananth, Miranda Christ, Yevgeniy Dodis, and Sam Gunn. Ideal pseudorandom codes. In *STOC*, 2025. arXiv:2411.05947. 4, 11

[BM09] Boaz Barak and Mohammad Mahmoody-Ghidary. Merkle puzzles are optimal - an $O(n^2)$-query attack on any key exchange from a random oracle. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 374–390, Santa Barbara, CA, USA, August 16–20, 2009. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-03356-8_22. 6

[CG24] Miranda Christ and Sam Gunn. Pseudorandom error-correcting codes. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part VI*, volume 14925 of *Lecture Notes in Computer Science*, pages 325–347, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland. doi:10.1007/978-3-031-68391-6_10. 3, 4, 5, 10, 11

[CT01] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 2001. doi:10.1002/0471200611. 9

[DMR25] Nico Döttling, Anne Müller, and Mahesh Sreekumar Rajasree. Separating pseudorandom codes from local oracles. *IACR Cryptol. ePrint Arch.*, page 1020, 2025. to appear at TCC 2025. URL: https://eprint.iacr.org/2025/1020. 4

[GG24] Surendra Ghentiyala and Venkatesan Guruswami. New constructions of pseudorandom codes. Cryptology ePrint Archive, Paper 2024/1425, 2024. URL: https://eprint.iacr.org/2024/1425. 3, 4, 5

[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. URL: http://papers.nips.cc/paper_files/paper/2024/hash/24c53bfa5b53fc2cf05644f5a7a26bb0-Abstract-Conference.html. 3, 4

[GZS25] Sam Gunn, Xuandong Zhao, and Dawn Song. An undetectable watermark for generative image models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL: https://openreview.net/forum?id=jlhBFm7T2J. 3

[HLLL25] Xuming Hu, Hanqian Li, Jungang Li, and Aiwei Liu. Videomark: A distortion-free robust watermarking framework for video diffusion models. *CoRR*, abs/2504.16359, 2025. URL: https://doi.org/10.48550/arXiv.2504.16359, arXiv:2504.16359, doi:10.48550/ARXIV.2504.16359. 3

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *21st Annual ACM Symposium on Theory of Computing*, pages 44–61, Seattle, WA, USA, May 15–17, 1989. ACM Press. doi:10.1145/73007.73012. 4, 6

[KKL88] Jeff Kahn, Gil Kalai, and Nathan Linial. The influence of variables on Boolean functions (extended abstract). In *29th Annual Symposium on Foundations of Computer Science*, pages 68–80, White Plains, NY, USA, October 24–26, 1988. IEEE Computer Society Press. doi:10.1109/SFCS.1988.21923. 12

[LMW25] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Black box crypto is useless for doubly efficient PIR. 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 VI*, volume 15606 of *Lecture Notes in Computer Science*, pages 65–93. Springer, 2025. doi:10.1007/978-3-031-91095-1_3. 3, 4, 8, 13, 14

[O'D14] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014. URL: http://www.cambridge.org/de/academic/subjects/computer-science/algorithmics-complexity-computer-algebra-and-computational-g/analysis-boolean-functions. 12

[RTV04] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20, Cambridge, MA, USA, February 19–21, 2004. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-540-24638-1_1. 4

[YZC+25] Zijin Yang, Xin Zhang, Kejiang Chen, Kai Zeng, Qiyi Yao, Han Fang, Weiming Zhang, and Nenghai Yu. Gaussian shading++: Rethinking the realistic deployment challenge of performance-lossless image watermark for diffusion models. *CoRR*, abs/2504.15026, 2025. URL: https://doi.org/10.48550/arXiv.2504.15026, arXiv:2504.15026, doi:10.48550/ARXIV.2504.15026. 3