On Robust Combiners for Pseudorandom Codes and PKE with Pseudorandom Ciphertexts

Nico Döttling[*1], Venkata Koppula[†2], Mahesh Sreekumar Rajasree[3], and Felix Rohrbach[1]

[1]CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
{doettling, felix.rohrbach}@cispa.de
[2]Indian Institute of Technology Delhi, New Delhi, India
kvenkata@iitd.ac.in
[3]University of Warsaw, Warsaw, Poland
srmahesh1994@gmail.com

Abstract

Robust combiners are a standard way to hedge cryptographic assumptions: given several candidate constructions, they produce one construction that remains secure as long as one candidate is secure. In favorable cases, such combiners also yield universal constructions. We ask whether this paradigm applies to cryptographic primitives whose public outputs must be both pseudorandom and sparse.

Our first result gives a negative answer for pseudorandom codes (PRCs), i.e., error-correcting codes whose codewords are computationally indistinguishable from uniform strings. We prove that secret-key PRCs admit no black-box robust combiner. The impossibility holds even for the weakest meaningful guarantee, where the combined scheme is required to be secure when all but one candidate PRC are secure.

We then show that the obstruction is not specific to error correction. In the sparse ciphertext regime, public-key encryption with pseudorandom ciphertexts also admits no black-box robust combiner. This stands in contrast to ordinary IND-CPA encryption, for which robust combiners are easy to obtain. Thus the requirement that public outputs look uniformly random introduces a genuine barrier to black-box assumption hedging, and rules out the standard combiner route to universal constructions for these primitives.

Keywords. robust combiners, pseudorandom codes, black-box separation, pseudorandom ciphertexts

1 Introduction

Cryptographic theory is still far from basing foundational primitives on minimal worst-case complexity assumptions, such as $\mathsf{BPP} \neq \mathsf{NP}$, even for one-way functions [BT03, AGGM06, AGGM10, BB15].

[*]Funded by the European Union (ERC, LACONIC, 101041207). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

[†]Funded by the HUJI-IITD Joint Global Research Partnership Program (IRD Project No. IR00016G_FS).

With current techniques, the optimistic state of affairs is often more modest: for a given primitive, one may have many plausible candidates, built from different assumptions and different mathematical concepts, without knowing which candidate, if any, is secure. A natural way to hedge this uncertainty is to use a *robust combiner*: given several candidate implementations of a primitive, the combiner outputs one new implementation that is secure as long as at least one of the candidates is secure, even if the remaining candidates are completely broken.

Combiners have become a general methodology for assumption hedging. They are known for basic primitives such as one-way functions and pseudorandom generators [HKN+05a], for encryption and key encapsulation mechanisms [HKN+05a, GHP18], and for stronger encryption notions such as chosen-ciphertext security [DK05, ZCW+16, SBGM25]. They also appear for more expressive objects, including functional encryption [ABJ+19, JMS20] and indistinguishability obfuscation [AJN+16, FHNS16, AJS17]. This line of work shows that, for many primitives, one need not identify the single right assumption in advance: a carefully designed combiner can preserve security as long as one of the proposed assumptions survives.

In some settings, combiners provide more than a way to combine a fixed list of candidates. They lead to *universal constructions*. If all polynomial-time candidate constructions of a primitive can be enumerated and then robustly combined, one obtains an explicit construction that is secure whenever any secure construction of the primitive exists. Levin's universal one-way function [Lev88] is the classical example of this phenomenon, and the connection between robust combiners and universal constructions is also central in the work of Ananth et al. [AJN+16] on indistinguishability obfuscation and witness encryption. Thus, for a primitive with many competing candidates, the existence of robust combiners can turn a non-constructive or uncertain existence landscape into a single canonical candidate.

This universalization viewpoint is especially compelling for primitives whose known constructions rely on specialized assumptions. Pseudorandom codes (PRCs), introduced by Christ and Gunn [CG24], are one such primitive. A PRC is an error-correcting code with a secret key: a legitimate decoder holding the key can recover the encoded message even after noise, while an observer without the key cannot distinguish codewords from uniformly random strings. PRCs were introduced in the context of watermarking AI models [ZDR19, KJGR21, Aar22, KGW+23, FGJ+25, ZALW24, KTHL24], where one wants a mark to be robust to perturbations but hidden from parties that do not know the secret key.

Known constructions of PRCs rely on assumptions with substantial coding-theoretic content, such as variants related to the McEliece problem [McE78], Learning Parity with Noise [Kea93, BFKL93], planted XOR and planted hyperloop assumptions [BKR23], local weak pseudorandom functions [GM24a], and permuted-code assumptions [CGG+25]. Recent works [DMSR26, GGW25] further show that PRCs cannot be constructed in a black-box way from broad classes of standard cryptographic primitives, even including virtual black-box obfuscation. These results make the following question natural: if PRCs cannot be generically built from standard primitives, can one at least obtain a universal PRC by robustly combining all candidate PRC constructions?

At first glance, this hope is not unreasonable. Pseudorandomness by itself does not preclude robust combination. For example, pseudorandom functions imply one-way functions, one-way functions admit robust combination, and one-way functions in turn imply pseudorandom functions via the standard HILL/GGM route [HILL99, GGM84]. Thus the obstruction, if any, cannot simply be that the primitive has a pseudorandomness requirement.

The primitives studied in this work have a more specific feature: they produce public objects

that must look uniformly random, while still carrying hidden structure that a legitimate party can recognize, decode, or decrypt. PRC codewords must look random, yet the secret decoder must recognize noisy codewords. Public-key encryption with pseudorandom ciphertexts has a similar flavor: ciphertexts should be indistinguishable from uniform strings, yet the secret key should decrypt them correctly. In both cases, the public objects carrying the hidden structure are sparse: under a fixed key, decodable or decryptable public outputs occupy only a negligible fraction of the ambient string space, even though sampled outputs must look uniform to an outsider. This leads to the central question of this paper:

Can cryptographic primitives with sparse pseudorandom public outputs be robustly combined in a black-box way?

Equivalently, can robust combiners provide universal constructions for such primitives?

1.1 Our Results

We answer this question negatively for two central primitives in which the decodable or decryptable public outputs are sparse, yet sampled outputs must be pseudorandom.

Our first result rules out black-box robust combiners for secret-key pseudorandom codes. We show that, in a natural oracle model, no polynomial-query combiner can take several candidate PRCs and produce a PRC that is guaranteed to be secure whenever at least one candidate is secure. In fact, the impossibility already holds for the weakest robustness requirement: the combiner must tolerate only one insecure candidate.

Theorem 1.1 (Informal)**.** *Black-box robust combiners for secret-key pseudorandom codes do not exist.*

Conceptually, this rules out the standard route to a universal PRC. A universal construction obtained by enumerating candidate PRCs and combining them would require precisely the kind of black-box robustness that our theorem excludes. Thus confidence in PRC security cannot be obtained by a generic assumption-hedging mechanism of this form; any successful combination must use additional structure of the candidate schemes in a non-black-box way.

Our second result shows that the same phenomenon is not limited to pseudorandom error-correcting codes. We also consider public-key encryption schemes whose ciphertexts are required to be pseudorandom, i.e., computationally indistinguishable from uniformly random strings. In the sparse ciphertext regime, where valid ciphertexts under a fixed public key form a negligible fraction of the ambient ciphertext space, we prove that black-box robust combiners are impossible.

Theorem 1.2 (Informal)**.** *Black-box robust combiners for public-key encryption schemes with pseudorandom ciphertexts in the sparse ciphertext regime do not exist.*

The sparsity condition is essential to this public-key statement. In dense ciphertext spaces, simple secret-sharing style combiners can preserve pseudorandomness, as we discuss in Section 2.3. Likewise, in the secret-key setting, pseudorandom ciphertext encryption admits simple robust combiners via the PRF combiner route; see Section 2.2. The impossibility therefore identifies a specifically public-key barrier in the sparse regime, rather than a general failure of pseudorandom ciphertexts to combine.

This also separates pseudorandom-ciphertext security from ordinary IND-CPA security. For standard public-key encryption, robust combiners are easy to obtain. Our result shows that this robustness disappears once ciphertexts must look uniformly random. In this sense, pseudorandom-ciphertext security is not just a syntactic strengthening of indistinguishability security; it is qualitatively more fragile from the viewpoint of black-box robust combination.

A simple corollary of our theorem, which to the best of our knowledge does not appear in the literature, is that there is no generic black-box transformation from ordinary IND-CPA public-key encryption to public-key encryption with pseudorandom ciphertexts. If such a transformation existed, one could first apply a standard robust combiner for IND-CPA encryption to candidate schemes and then transform the result into a pseudorandom-ciphertext scheme. This would give a robust combiner for pseudorandom-ciphertext public-key encryption, contradicting our impossibility theorem in the sparse regime.

Corollary 1.3 (Informal)**.** *There is no black-box transformation from IND-CPA secure public-key encryption to public-key encryption with pseudorandom ciphertexts in the sparse ciphertext regime.*

The common message of our two impossibility results is that pseudorandomness alone is not the barrier. Rather, the difficulty arises when pseudorandom-looking public objects must also contain sparse hidden structure that enables legitimate decoding or decryption. This combination prevents the usual black-box combiner paradigm from yielding universal constructions.

2 Technical Overview

We begin the technical overview with some preliminaries on (secret key) pseudorandom codes and black-box combiners.

Secret key pseudorandom codes. Let $\tau \in [0,1]$. A τ-robust secret-key PRC scheme for message space $\mathcal{M}$ consists of two algorithms - a randomized encoding algorithm Enc, and a decoding algorithm Dec. Without loss of generality, we will assume that the secret key space is $\{0,1\}^\ell$, randomness space is $\{0,1\}^r$ and the code space is $\{0,1\}^n$ (where ℓ, r, n are polynomial in the security parameter). The encoding algorithm takes as input a secret key sk, a message $m \in \mathcal{M}$, and outputs an encoding $c \in \{0,1\}^n$. The decoding algorithm takes as input a secret key sk, a string $x \in \{0,1\}^n$, and outputs $y \in \mathcal{M} \cup \{\perp\}$. For correctness, we require that for all secret keys sk, for all messages m, if $c \leftarrow \mathsf{Enc}(\mathsf{sk}, m)$, then $\mathsf{Dec}(\mathsf{sk}, c') = m$ for all c' that are τ-close to c (that is, the Hamming distance between c and c' is at most $\tau \cdot n$). For security, we require that for every polynomial time adversary $\mathcal{A}$, the encoding of adversarially chosen messages is computationally indistinguishable from uniformly random strings in $\{0,1\}^n$.

In this work, we restrict our attention to zero-bit secret-key PRCs. Here, the encoding algorithm takes as input the secret key sk (but no message), and outputs a codeword $c \in \{0,1\}^n$. The decoding algorithm takes as input a secret key sk, a string $x \in \{0,1\}^n$, and outputs either $\top$ or $\perp$. Correctness and security are similar to regular PRCs. Christ and Gunn [CG24] showed that any zero-bit PRC scheme can be upgraded to a PRC scheme for message space $\mathcal{M}$.

Black box combiners. A cryptographic combiner is a mechanism that takes multiple, independent implementations of a specific primitive (such as pseudorandom codes in our case) and unifies them

into a single, composite construction. The central objective of a combiner is to guarantee security as long as a minimum threshold of the underlying candidate implementations remains secure.

More formally, let $0 < k < n$. An $(n, n\text{-}k)$ combiner for PRCs takes as input n PRC candidates $\mathsf{PRC}_1, \mathsf{PRC}_2, \ldots, \mathsf{PRC}_n$ and produces a combined PRC candidate $\mathsf{PRC}_{\mathsf{cmb}}$. This combined scheme is secure if at least n-k of the PRC candidates are secure.

A black-box combiner is one where the combiner algorithms $\mathsf{Enc}_{\mathsf{cmb}}$ and $\mathsf{Dec}_{\mathsf{cmb}}$ only make black-box use of the n candidates. As a result, given any secret key sk and randomness r, $\mathsf{Enc}_{\mathsf{cmb}}(\mathsf{sk}; r)$[1] can be computed using just oracle access to $\mathsf{PRC}_1, \ldots, \mathsf{PRC}_n$. Similarly, for any sk and y, $\mathsf{Dec}_{\mathsf{cmb}}(\mathsf{sk}, y)$ can be computed using only oracle access to the PRC candidates.

2.1 Impossibility of Combiners for Pseudorandom Codes

In this work, we show that $(n, n\text{-}1)$ black-box combiners do not exist for pseudorandom codes. In order to show a black-box impossibility, it suffices to show that for any combiner, there exists an unbounded adversary $\mathcal{A}$ that succeeds in the following experiment. First, the adversary $\mathcal{A}$, depending on the combiner's structure, chooses an index $i \in [n]$. All PRC schemes are *perfectly secure* — even an unbounded adversary cannot break the scheme's security if it is allowed only polynomially many queries to the encoder and decoder. However, the adversary receives the full description of the PRC scheme PRC_i, thereby corrupting the i-th oracle. The adversary now participates in the PRC security game with $(\mathsf{Enc}_{\mathsf{cmb}}^{\mathsf{PRC}_1,\ldots,\mathsf{PRC}_n}, \mathsf{Dec}_{\mathsf{cmb}}^{\mathsf{PRC}_1,\ldots,\mathsf{PRC}_n})$.

For ease of exposition, we present our impossibility result for $(2, 1)$ black-box PRC combiners. That is, one of the candidates is adversarially chosen, and the other candidate is perfectly secure. The perfectly secure candidate is obtained by sampling a uniformly random function $f : \{0,1\}^\ell \times \{0,1\}^r \to \{0,1\}^n$, which represents the encoding function. We choose n sufficiently larger than $\ell + r$, say $n \geq \ell + r + \omega(\log \lambda)$, so that the Hamming balls around the random codewords are almost disjoint and occupy only a negligible fraction of $\{0,1\}^n$ [2]. The decoder, on input sk, y, checks if there is a $\rho \in \{0,1\}^r$ such that y is $\tau \cdot n$ close to $f(\mathsf{sk}, \rho)$. We call this the *random PRC model*, and this is similar to how the random oracle is used when separating one-way functions from other cryptographic primitives (such as public key encryption).

The High-Level Structure of the Impossibility. Our impossibility result starts with a simple observation: if the combiner decoder $\mathsf{Dec}_{\mathsf{cmb}}$ does not query the underlying PRC schemes at all, then the combiner is immediately vulnerable to an unbounded distinguisher. Suppose the distinguisher receives challenge words $cw_0^\star, \ldots, cw_k^\star$ corresponding to known messages $m_0, \ldots, m_k$. The challenge is either the real distribution, where each $cw_i^\star$ is an honestly generated combiner codeword for m_i, or the random distribution, where the $cw_i^\star$'s are uniformly random strings of the appropriate length.

The distinguisher enumerates all possible combiner secret keys sk and checks how many of the tests $\mathsf{Dec}_{\mathsf{cmb}}(\mathsf{sk}, cw_i^\star) = m_i$ are satisfied. In the real case, by correctness, the true secret key satisfies most of these tests with overwhelming probability[3]. In the random case, however, a uniformly

[1] Recall, we are working with zero-bit PRCs, and therefore the encoding algorithm only takes the secret key and randomness as input.

[2] For zero bit PRCs, one also usually requires a soundness property: for every string y, only a negligible fraction of secret keys sk should satisfy $\mathsf{Dec}(\mathsf{sk}, y) = \top$. Such a property requires the codeword space to be sufficiently larger than the total number of codewords.

[3] If the combined scheme satisfies perfect correctness, then all the tests are satisfied. If it has a small correctness error, then with overwhelming probability, most of the tests are satisfied.

random string is accepted as a valid encoding of the prescribed message m_i under any fixed key only with negligible probability in the sparse-codeword regime. Hence, even after taking a union bound over all possible secret keys, with overwhelming probability no key satisfies many of the tests. Therefore, the distinguisher can separate real combiner codewords from uniformly random strings, showing that a query-free decoder is susceptible to a direct attack on the pseudorandomness property.

It remains to handle the general case, where the combiner decoder may make oracle queries to the underlying PRCs. The proof proceeds by looking at the *first* oracle query made by the decoder on a challenge word. There are only two possibilities: the first query is either a *decode* query or an *encode* query to one of the underlying PRC schemes. We show that in both cases this first query can be removed, at the cost of only a small loss in correctness and security. Repeating this argument query by query eventually transforms the original decoder into an augmented decoder that makes no oracle queries at all. The oracle-free attack described above then breaks the pseudorandomness of the combiner, giving the desired contradiction.

1. First, we handle the case where the first oracle query made by the combiner decoder is a *decode* query. We show that such a query is essentially useless. More precisely, if the decoder first queries one of the branch decoders and receives either $\top$ or $\bot$, then with overwhelming probability this answer is $\bot$, even when the input is an honestly generated combiner codeword. Thus one can replace the first decode query by the fixed answer $\bot$, while losing only a small amount of correctness.

2. Second, we handle the case where the first oracle query is an *encode* query. This case is more delicate, because the answer to an encode query may be correlated with the challenge codeword: the same branch encoding entry may already have been used by $\mathsf{Enc}_{\mathsf{cmb}}$ while producing the codeword. Here we use the heavy-query and exceptionality machinery of [GGW25, DM24] as a black box. It says that all globally heavy encode queries can be learned in advance and stored in an augmented key, while globally light queries can be locally simulated except with small probability.

Why Decode Queries Are Useless. We begin with the case where the first oracle query made by $\mathsf{Dec}_{\mathsf{cmb}}$ is a decode query. For a combiner key sk and input x, let

$$\mathsf{Qry}_{\mathsf{sk}}(x; \rho_{\mathsf{Qry}}) = (b, \kappa, cw_0)$$

denote the first query generated by the combiner decoder where ρ_{Qry} is the randomness. The bit $b \in \{0,1\}$ specifies which branch decoder is queried, and the combiner then receives the answer $\mathsf{Dec}_b(\kappa, cw_0) \in \{\top, \bot\}$.

For each branch $j \in \{0,1\}$, define two probabilities. The first is the probability that, on a real combiner codeword, the first query goes to branch j and receives answer $\bot$:

$$p^{\mathrm{real}}_{\mathsf{sk},j} = \Pr_{c \leftarrow \mathsf{Enc}_{\mathsf{cmb}}(\mathsf{sk}),\, \rho_{\mathsf{Qry}}} \left[b = j \wedge \mathsf{Dec}_j(\kappa, cw_0) = \bot \mid (b, \kappa, cw_0) = \mathsf{Qry}_{\mathsf{sk}}(c; \rho_{\mathsf{Qry}}) \right].$$

The second is the analogous probability when the input to the combiner decoder is uniformly random:

$$p^{\mathrm{rand}}_{\mathsf{sk},j} = \Pr_{u \leftarrow \{0,1\}^{n_{\mathsf{cmb}}},\, \rho_{\mathsf{Qry}}} \left[b = j \wedge \mathsf{Dec}_j(\kappa, cw_0) = \bot \mid (b, \kappa, cw_0) = \mathsf{Qry}_{\mathsf{sk}}(u; \rho_{\mathsf{Qry}}) \right].$$

The first main observation is a transfer statement. If, for a noticeable fraction of keys sk, the two probabilities $p^{\mathrm{real}}_{\mathsf{sk},j}$ and $p^{\mathrm{rand}}_{\mathsf{sk},j}$ differ noticeably, then one can break the pseudorandomness of the combiner. Indeed, an unbounded adversary corrupts branch j, obtains the complete truth table of Dec_j, samples many challenge outputs from the combiner pseudorandomness game, and then brute-forces over all candidate combiner keys sk. For each candidate key, it computes the empirical frequency with which the first query goes to branch j and receives $\perp$.

In the random world, where the challenge oracle returns independent uniform strings, this empirical frequency concentrates around $p^{\mathrm{rand}}_{\mathsf{sk},j}$ for every candidate key simultaneously. In the real world, where the challenge oracle returns outputs of $\mathsf{Enc}_{\mathsf{cmb}}(\mathsf{sk}^\star)$, the empirical frequency for the true key $\mathsf{sk}^\star$ concentrates around $p^{\mathrm{real}}_{\mathsf{sk}^\star,j}$. Therefore, any noticeable gap between the real and random first-query behavior gives an unbounded distinguisher against the combiner. Since the combiner is assumed pseudorandom even when branch j is corrupted, such a gap can occur only for a negligible fraction of keys.

Sparse Acceptance of a Random Branch Decoder The second ingredient is purely information-theoretic. In the random PRC model, for a fixed key κ, the accepting region of the branch decoder is

$$A^{(j)}_{\kappa} = \bigcup_{r\in\{0,1\}^{r_j}} B(C^{(j)}_{\kappa,r}, \tau n_j),$$

where $B(C^{(j)}_{\kappa,r}, \tau n_j)$ is the Hamming ball of radius τn_j around the random codeword $C^{(j)}_{\kappa,r}$.

We set $n_j = \lambda^2$ and $r_j = \lambda$, the relative volume of this accepting region is at most

$$2^{r_j} \cdot \frac{\mathsf{Vol}_{n_j}(\tau n_j)}{2^{n_j}} \le 2^{\lambda-(1-H(\tau))\lambda^2} = 2^{-\Omega(\lambda^2)}.$$

Thus, for a uniformly random word cw_0, the probability that $\mathsf{Dec}_j(\kappa, cw_0)$ accepts is exponentially small.

This remains true even when (κ, cw_0) is generated by an arbitrary randomized first-query function $\mathsf{Qry}_{\mathsf{sk}}(u;\rho_{\mathsf{Qry}})$ on a uniformly random input u. Averaging over the lazy sampling of the branch PRC and then using Markov's inequality, we get that, except with overwhelming probability over the sampled branch oracle, for all but a negligible fraction of combiner keys sk, the random-world acceptance mass

$$a^{\mathrm{rand}}_{\mathsf{sk},j} = \Pr_{u,\rho_{\mathsf{Qry}}}\left[b = j \wedge \mathsf{Dec}_j(\kappa, cw_0) \neq \perp \mid (b,\kappa,cw_0) = \mathsf{Qry}_{\mathsf{sk}}(u;\rho_{\mathsf{Qry}})\right]$$

is negligible. In short, on a uniformly random input to the combiner decoder, the first branch decode query almost always receives $\perp$.

Transferring Sparse Acceptance to Real Codewords. Now we combine the two observations. Sparse acceptance says that, in the random world, $a^{\mathrm{rand}}_{\mathsf{sk},0} + a^{\mathrm{rand}}_{\mathsf{sk},1}$ is negligible for almost all combiner keys. Equivalently, $p^{\mathrm{rand}}_{\mathsf{sk},0} + p^{\mathrm{rand}}_{\mathsf{sk},1} \approx 1$. The transfer theorem says that, for each branch j, pseudorandomness of the combiner forces $p^{\mathrm{real}}_{\mathsf{sk},j} \approx p^{\mathrm{rand}}_{\mathsf{sk},j}$ for almost all keys. Therefore, $p^{\mathrm{real}}_{\mathsf{sk},0} + p^{\mathrm{real}}_{\mathsf{sk},1} \approx 1$.

But the left-hand side is exactly the probability that the first decode query made by $\mathsf{Dec}_{\mathsf{cmb}}$ on a real combiner codeword receives the answer $\perp$. Hence, with overwhelming probability, the first branch decode query is answered by $\perp$ even on honestly generated codewords.

This allows us to define an oracle-free simulation for the first decode query: run $\mathsf{Dec}_{\mathsf{cmb}}$ until it makes its first branch decode query, answer this query by $\perp$, and then continue the simulation. Since the real execution already receives $\perp$ with probability close to one, this replacement decreases correctness by only a small additive amount.

Why Encode Queries Require a Different Argument. A first encode query is fundamentally different. Suppose the first query is $\mathsf{Qry}_{\mathsf{sk}}(x; \rho_{\mathsf{Qry}}) = (b, \kappa, r)$, and the answer is $cw_0 = \mathsf{Enc}_b(\kappa; r)$. Unlike a decode query, this answer is not usually $\perp$ and cannot simply be replaced by a fixed value.

The key difficulty is correlation with the combiner encoder. While producing a combiner codeword c, the encoder $\mathsf{Enc}_{\mathsf{cmb}}$ may itself have queried one of the branch encoders on exactly the same tuple (b, κ, r). If the decoder later asks this same query, then the correct answer may be correlated with the codeword c. Locally simulating the answer by a fresh random string would then be inconsistent with the encoder transcript.

Thus, encode queries are not useless in the same direct sense as decode queries. Instead, they must be compiled out.

Heavy Encode Queries. For a fixed combiner key sk, define the global weight of an encode query $q = (b, \kappa, r)$ by

$$w_{\mathsf{sk}}(q) = \Pr_{u, \rho_{\mathsf{Qry}}} [\mathsf{Qry}_{\mathsf{sk}}(u; \rho_{\mathsf{Qry}}) = q] ,$$

where u is uniformly random in the combiner codeword space.

A query is globally heavy if $w_{\mathsf{sk}}(q) \geq \varepsilon$. Since the global weights sum to one, there are at most $1/\varepsilon$ globally heavy queries. An unbounded learner can therefore find all of them, query the corresponding branch encoders, and store the query-response pairs in an augmented secret key.

The augmented decoder then handles a first encode query as follows. If the query is globally heavy, it uses the stored answer. If the query is globally light, it locally simulates the answer by a fresh uniformly random codeword.

This simulation is perfect unless the query is both globally light and was already used by the honest combiner encoder while generating the codeword. Such a query is an intersection query between the encoder and decoder transcripts. Bounding the probability of this bad event is the only remaining task.

Local Weights and Exceptional Codewords To control the bad event, we use the framework of [DMSR26]. For a fixed codeword c, define the local weight of a query q around c by

$$\ell_{\mathsf{sk},c}(q) = \Pr_{e, \rho_{\mathsf{Qry}}} [\mathsf{Qry}_{\mathsf{sk}}(c + e; \rho_{\mathsf{Qry}}) = q] ,$$

where e is Bernoulli noise.

A codeword c is called (ε, δ)-exceptional if there exists a query q such that $w_{\mathsf{sk}}(q) < \varepsilon$ but $\ell_{\mathsf{sk},c}(q) \geq \delta$. Thus, exceptional codewords are precisely those around which some globally light query becomes locally heavy.

The heavy-query theorem from [DMSR26] says that exceptional points are rare under the uniform distribution. Moreover, because exceptionality can be tested from the query function itself and does not require oracle access, pseudorandomness of the combiner transfers this rarity from uniform strings to real combiner codewords. Therefore, for a real codeword $c \leftarrow \mathsf{Enc}_{\mathsf{cmb}}(\mathsf{sk})$, the event that c is exceptional occurs only with small probability.

Now condition on c being non-exceptional. Every globally light query has local weight less than δ around c. The honest combiner encoder makes at most $M_{\mathsf{cmb}}(\lambda)$ branch encode queries while producing c. Hence, by a union bound, the probability that the noisy decoder query hits one of these globally light encoder queries is at most $M_{\mathsf{cmb}}(\lambda) \cdot \delta$. Choosing δ and ε appropriately makes this probability as small as any desired inverse polynomial.

Thus, after storing all globally heavy encode queries, the first encode query can also be compiled out with only a small loss in correctness.

Removing All Branch Oracle Queries. The previous two steps show how to eliminate an oracle query made by the combiner decoder, depending on whether the query is an encode query or a decode query. For each query, we proceed as follows.

- For an encode query, we augment the key with the responses to all globally heavy queries, and locally simulate the responses to globally light queries.
- For a decode query, we first check whether the queried codeword is covered by the information stored in the augmented key or was previously simulated during an encode query. If so, we return $\top$; otherwise, we return $\bot$.

The full proof applies these transformations iteratively. Each step removes one oracle query from the decoder and incurs only a controlled additive loss in correctness. Since the combiner decoder makes only polynomially many branch oracle queries, we choose the loss per step small enough so that the total loss remains inverse-polynomial. At the end of the process, we obtain an augmented decoder $\mathsf{ADec}_{\mathsf{cmb}}$ that no longer makes any oracle queries to the branch PRCs but still accepts honestly generated noisy combiner codewords with noticeable probability.

2.2 Existence of Combiners for Pseudorandom SKE

It is useful to contrast the PRC setting with the much simpler situation for secret-key encryption with pseudorandom ciphertexts. In the IND$-CPA SKE setting, robust combination is straightforward. The reason is that a secure secret-key encryption scheme already gives enough pseudorandomness to build standard symmetric-key primitives.

Concretely, let $\mathsf{SKE}_0 = (\mathsf{Enc}_0, \mathsf{Dec}_0)$ and $\mathsf{SKE}_1 = (\mathsf{Enc}_1, \mathsf{Dec}_1)$ be two candidate secret-key encryption schemes, and suppose that at least one of them is IND$-CPA-secure. From each candidate scheme, we derive a keyed pseudorandom function family $f_{k_i} : \{0,1\}^\ell \to \{0,1\}^{|m|}$ using the standard route from one-way functions to pseudorandom generators and then to pseudorandom functions [HILL99]. Intuitively, if SKE_i is secure, then it provides the computational hardness needed to obtain a secure PRF f_{k_i}.

The combiner encryption algorithm is then defined in the most direct way. To encrypt a message m, sample independent random strings r_0, r_1, and output

$$\mathsf{ct} = \big(r_0,\, r_1,\, f_{k_0}(r_0) \oplus f_{k_1}(r_1) \oplus m\big).$$

The decryption algorithm recomputes both PRF values and recovers $m = f_{k_0}(r_0) \oplus f_{k_1}(r_1) \oplus \mathsf{ct}_3$, where ct_3 denotes the third component of the ciphertext.

Correctness is immediate. For security, observe that if at least one of the two candidate schemes is secure, then at least one of the two PRFs is pseudorandom. Therefore, the value $f_{k_0}(r_0) \oplus f_{k_1}(r_1)$

is pseudorandom even if the other PRF is completely insecure. Hence the third component of the ciphertext is a one-time-pad encryption of m under a pseudorandom pad, and the full ciphertext is pseudorandom. This gives a simple $(2,1)$-robust combiner for IND$-CPA SKE. In fact, the same idea extends immediately to an $(n,1)$-robust combiner: encrypt by XORing the message with the XOR of n independently generated PRF outputs, one derived from each candidate scheme. As long as at least one candidate scheme is secure, at least one PRF output is pseudorandom, and therefore the entire XOR pad remains pseudorandom.

2.3 Existence of Combiners for Pseudorandom PKE with Dense Ciphertext Space

In this subsection, we explain why pseudorandom public-key encryption admits simple robust combiners when the *underlying candidate schemes* have dense ciphertext spaces. In the ideal dense case, for every public key p, the encryption map

$$e(p,\cdot,\cdot) : \{0,1\} \times \{0,1\}^{\ell^r} \to \{0,1\}^{\ell^{\mathsf{ct}}}$$

is a permutation, or close to a permutation, with $\ell^{\mathsf{ct}} = \ell^r + 1$. Then every ciphertext string is valid under p, and a uniformly random ciphertext decrypts to a uniformly random bit.

This is precisely the property that allows the standard secret-sharing combiner to preserve pseudorandomness. Let

$$\mathsf{PKE}_0 = (\mathsf{Keygen}_0, \mathsf{Enc}_0, \mathsf{Dec}_0), \qquad \mathsf{PKE}_1 = (\mathsf{Keygen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$$

be two candidate public-key encryption schemes. The combined key-generation algorithm independently generates $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{Keygen}_0$ and $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{Keygen}_1$, and outputs $\mathsf{pk} = (\mathsf{pk}_0, \mathsf{pk}_1)$ and $\mathsf{sk} = (\mathsf{sk}_0, \mathsf{sk}_1)$. To encrypt a bit b, choose a uniform share $a \leftarrow \{0,1\}$, compute $c_0 \leftarrow \mathsf{Enc}_0(\mathsf{pk}_0, a)$ and $c_1 \leftarrow \mathsf{Enc}_1(\mathsf{pk}_1, a \oplus b)$ and output $\mathsf{ct} = (c_0, c_1)$. The combined decryptor computes $a_0 := \mathsf{Dec}_0(\mathsf{sk}_0, c_0)$ and $a_1 := \mathsf{Dec}_1(\mathsf{sk}_1, c_1)$, and outputs $a_0 \oplus a_1$.

Correctness is immediate. For security, suppose branch 0 is corrupted. In the real world, decrypting c_0 reveals only the uniform share a. In the random world, c_0 is a uniformly random branch ciphertext string; density ensures that this string decrypts to a plausible uniform share rather than to $\perp$. The second component remains protected by the pseudorandomness of the uncorrupted branch. The same argument applies symmetrically when branch 1 is corrupted. Thus, in a dense IND$-CPA PKE-like model, the secret-sharing combiner preserves pseudorandomness for the same reason that the usual concatenation combiner preserves IND$-CPA security: corrupting one branch reveals only a random share, while the other share remains hidden.

There are also composition-style combiners. For example, one may first encrypt under an inner branch and then encrypt the resulting ciphertext under an outer branch:

$$c_{\mathrm{in}} \leftarrow \mathsf{Enc}_0(\mathsf{pk}_0, b), \qquad c_{\mathrm{out}} \leftarrow \mathsf{Enc}_1(\mathsf{pk}_1, c_{\mathrm{in}}),$$

and output c_{out}. The combined decryptor first decrypts the outer ciphertext to recover c_{in}, and then decrypts c_{in} using the inner branch.

For such a composition combiner, the exact density requirement depends on which layer is exposed. If the outer branch is corrupted, then a uniformly random outer ciphertext should decrypt to a plausible inner ciphertext rather than to $\perp$. Thus density of the outer branch is essential. Once a plausible inner ciphertext is obtained, the pseudorandomness of the inner branch can protect the message. On the other hand, if the inner branch is corrupted but the outer branch is secure, then the outer pseudorandom ciphertext already hides the entire inner ciphertext.

2.4 Impossibility of Combiners for Pseudorandom PKE with Sparse Ciphertext Space

We prove the impossibility of $(2,1)$-combiners for single bit public-key encryption with pseudorandom ciphertexts in an Impagliazzo–Rudich (IR) style framework. Here, the adversary first learns the oracle information that is likely to be shared between key generation and a random encryption instance, resamples a fake secret key consistent with this learned information, and then uses the fake key to decrypt the challenge ciphertext in a carefully simulated oracle world.

The IR template. Before describing the combiner impossibility, we recall the basic Impagliazzo–Rudich intuition for the simpler problem of showing a black-box separation between public-key encryption and one-way functions. To establish this separation, it suffices to show that for any oracle-aided PKE scheme $\mathsf{PKE}^{-} = (\mathsf{Keygen}^{-}, \mathsf{Enc}^{-}, \mathsf{Dec}^{-})$, there exists an unbounded adversary $\mathcal{A}$ such that for a uniformly random function $\mathcal{H}$, $\mathcal{A}$ can break the CPA security of $\mathsf{PKE}^{\mathcal{H}} = (\mathsf{Keygen}^{\mathcal{H}}, \mathsf{Enc}^{\mathcal{H}}, \mathsf{Dec}^{\mathcal{H}})$ by making only polynomially many queries to $\mathcal{H}$.

Fix any arbitrary oracle-aided public-key encryption scheme $\mathsf{PKE}^{\mathcal{H}} = (\mathsf{Keygen}^{\mathcal{H}}, \mathsf{Enc}^{\mathcal{H}}, \mathsf{Dec}^{\mathcal{H}})$, where each of $\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}$ makes at most $N = N(\lambda)$ oracle queries, and let us assume the message space is $\{0,1\}$. It is helpful to view a key-generation execution as producing not only $(\mathsf{pk}, \mathsf{sk})$, but also a transcript of oracle queries and answers. Let $K := \mathsf{View}^{\mathcal{H}}_{\mathsf{Keygen}}$ denote this key-generation view. Similarly, for a fresh encryption of a bit b under pk, let $B := \mathsf{View}^{\mathcal{H}}_{\mathsf{Enc}}(\mathsf{pk}, b)$ denote the corresponding encryption view. The public key pk is a public handle to information created during key generation. Since pk is visible to everyone, it cannot by itself contain a secret. Rather, correctness requires that pk guide the encryptor toward producing ciphertexts that are compatible with some hidden information retained in sk. In a random-oracle construction, such hidden compatibility can only arise through oracle information: key generation and encryption must, with non-negligible probability, depend on common oracle locations or on oracle values that are connected through the public key. This is the core IR intuition.

The public key pk is public, and all the algorithms are public. Hence, if a fresh encryption used only oracle queries that are independent of the key-generation view, then the ciphertext would be generated from public information and fresh randomness that has no hidden correlation with the secret key. Such a ciphertext cannot carry secret-key-dependent structure that is useful to the honest decryptor but hidden from an unbounded eavesdropper. Therefore, any security-relevant correlation between the ciphertext and the secret key must pass through oracle information that is seen, directly or indirectly, on both sides: by key generation when creating $(\mathsf{pk}, \mathsf{sk})$, and by encryption when producing the ciphertext. These shared oracle locations are the *intersection queries*.

The IR attack is based on learning the important intersection queries. Suppose Eve receives a public key pk. She can run the encryption algorithm herself, because encryption is public. She repeatedly encrypts (polynomially many times) both bits under pk, using fresh encryption randomness each time. During these trial encryptions, Eve records all oracle query-response pairs that the encryptor makes. Let I denote the set of oracle information learned in this way.

The purpose of this learning phase is not to learn every possible encryption query. That would be impossible, since the encryptor may have (exponentially) many possible random choices. Instead, Eve only needs to learn the queries that are likely to appear in a random encryption process. For a fixed public key pk and bit b, define the encryption weight of a query q by

$$p_b(q) := \Pr_r\big[q \in Q(\mathsf{View}^{\mathcal{H}}_{\mathsf{Enc}}(\mathsf{pk}, b; r))\big],$$

where the probability is over the fresh encryption randomness and $Q(\mathsf{View}^{\mathcal{H}}_{\mathsf{Enc}}(\mathsf{pk}, b; r))$ denotes the set of queries in the view. If $p_b(q)$ is large, then q is a *heavy encryption query*. Such a query is likely to appear in a fresh encryption of b, and therefore Eve will see it after sufficiently many trial encryptions. Since one encryption makes at most N oracle queries,

$$\sum_q p_b(q) = \mathbb{E}_r\left[|Q(\mathsf{View}^{\mathcal{H}}_{\mathsf{Enc}}(\mathsf{pk}, b; r))|\right] \le N.$$

Therefore there cannot be too many heavy queries. If we call a query q heavy when $p_b(q) \ge \tau$, then there are at most N/τ heavy queries. After m independent trial encryptions of bit b, a fixed heavy query is missed with probability at most $(1-\tau)^m \le e^{-m\tau}$. Taking a union bound over all heavy queries, Eve misses some heavy query with probability at most $\frac{N}{\tau}e^{-m\tau}$. Thus, by choosing $m = O\big(\frac{N}{\delta}\log\frac{N}{\delta}\big)$ and $\tau = \Theta\big(\frac{\delta}{N}\big)$, Eve learns all heavy encryption queries except with probability at most δ. Eve performs this learning separately for both bits because she does not know which bit will later be encrypted in the challenge ciphertext.

After this learning phase, Eve receives a challenge ciphertext $\mathsf{ct}^\star \leftarrow \mathsf{Enc}^{\mathcal{H}}(\mathsf{pk}, b^\star)$, where $b^\star \in \{0,1\}$ is hidden. Let $B^\star$ denote the actual challenge-encryption view. Eve would like to decrypt $\mathsf{ct}^\star$, but she does not know the real secret key. The IR idea is to replace the real secret key by a fake one.

Since Eve is computationally unbounded, she can sample exactly from the conditional distribution of key-generation views that are consistent with what she knows. Namely, she samples

$$K' \leftarrow \mathsf{Samp}_{\mathsf{Keygen}}(\mathsf{pk}, I),$$

where this means: sample an honest key-generation view, including its internal randomness, its oracle query-answer transcript, and the resulting secret key, conditioned on outputting the same public key pk and on agreeing with all oracle query-answer pairs in I.

This sampling does not require any additional access to the real oracle. The reason is that Eve is not trying to recover the real key-generation view. She is sampling a fresh hypothetical key-generation view from the correct conditional distribution. Since the random oracle has no structure beyond its input-output values, all oracle answers at locations outside I can be sampled internally as fresh random values, while answers at locations already in I are fixed to their learned values. Equivalently, an unbounded Eve may enumerate all possible key-generation random tapes and all possible oracle-answer transcripts of length at most N, retain exactly those that output pk and are consistent with I, and sample from this conditional space. Let sk' be the secret key contained in this fake key-generation view.

Surely, sk' need not be the real secret key. However, Eve does not need the real secret key. She only needs a secret key that is compatible with the challenge encryption. The only way the fake key-generation view K' can be incompatible with the challenge-encryption view $B^\star$ is if both views ask some common oracle query that Eve did not already learn. Indeed, if $Q(K') \cap Q(B^\star) \subseteq Q(I)$, then all common oracle answers between K' and $B^\star$ are already fixed inside Eve's learned transcript I. Since K' was sampled consistently with I, the fake key-generation view and the real challenge-encryption view agree on every oracle location where they overlap. This motivates the good event

$$\mathsf{Good}_{\mathsf{PKE}} \quad \Longleftrightarrow \quad Q(K') \cap Q(B^\star) \subseteq Q(I).$$

The heavy-query argument above implies that $\mathsf{Good}_{\mathsf{PKE}}$ holds with high probability. To see why, condition on Eve not missing any heavy encryption query. Then every query outside I that appears

in K' is light from the point of view of a fresh encryption. Since K' contains at most N queries, the probability that the challenge encryption hits any one of these remaining light queries is at most $N\tau$, which is small by the choice of τ. Thus Eve learns all relevant intersections except with probability at most δ.

It is important to note what Eve has and has not learned. The transcript I is not meant to contain the entire challenge-encryption view $B^\star$. It is only meant to contain the intersection queries between $B^\star$ and the fake key-generation view K'. Fresh encryption queries that do not belong to K' may remain completely unlearned. For example, an encryption algorithm may choose a fresh random string x, include the pair $(x, \mathcal{H}(x))$ in the ciphertext, and the decryptor may later verify this pair by querying $\mathcal{H}(x)$. Such a query x is light and will typically not be learned during Eve's sampling phase. This causes no inconsistency with the fake secret key, because $x \notin Q(K')$. Eve can therefore answer such non-intersection queries using the real oracle.

Now Eve decrypts using the fake key. Define a mixed oracle $\widetilde{\mathcal{H}}$ by answering according to the fake key-generation transcript on queries in K', and according to the real oracle elsewhere:

$$\widetilde{\mathcal{H}}(q) = \begin{cases} K'(q), & q \in Q(K'), \\ \mathcal{H}(q), & q \notin Q(K'). \end{cases}$$

Eve runs $\mathsf{Dec}^{\widetilde{\mathcal{H}}}(\mathsf{sk}', \mathsf{ct}^\star)$. On the event $\mathsf{Good}_{\mathsf{PKE}}$, the challenge encryption view $B^\star$ is still a valid encryption view under the mixed oracle $\widetilde{\mathcal{H}}$. Indeed, each query of $B^\star$ either lies outside K', in which case $\widetilde{\mathcal{H}}$ agrees with the real oracle $\mathcal{H}$, or lies inside K', in which case it is an intersection query already contained in I, where K' was forced to agree with $\mathcal{H}$. Therefore $B^\star$ produces the same ciphertext $\mathsf{ct}^\star$ under $\widetilde{\mathcal{H}}$.

Similarly, K' is by construction a valid key-generation view under $\widetilde{\mathcal{H}}$, producing $(\mathsf{pk}, \mathsf{sk}')$. Hence, on $\mathsf{Good}_{\mathsf{PKE}}$, the three objects $\big(K', B^\star, \mathsf{View}^{\widetilde{\mathcal{H}}}_{\mathsf{Dec}}(\mathsf{sk}', \mathsf{ct}^\star)\big)$ can be embedded into one consistent random-oracle execution: first key generation produces $(\mathsf{pk}, \mathsf{sk}')$, then encryption of $b^\star$ produces $\mathsf{ct}^\star$, and finally decryption is run using sk'. By correctness of the PKE scheme, decryption outputs $b^\star$ with probability close to the honest correctness probability, except on the bad event $\neg\mathsf{Good}_{\mathsf{PKE}}$.

Thus Eve, using only polynomially many real oracle queries, can recover the challenge bit with probability essentially equal to the correctness probability of the scheme. If the scheme has overwhelming correctness, then Eve guesses $b^\star$ with probability close to one, contradicting IND-CPA security.

Random labelled-PKE oracle model and cells. To show our impossibility result for IND$-CPA PKE combiners, we will use the same IR framework, but the underlying oracle is no longer a plain random oracle. Instead, the construction has black-box access to two independent random labelled-PKE oracles

$$\mathsf{PKE}_0 = (g_0, e_0, d_0), \qquad \mathsf{PKE}_1 = (g_1, e_1, d_1).$$

For each branch $j \in \{0, 1\}$, the oracle g_j maps a branch secret key s to a branch public key $p = g_j(s)$. The oracle e_j encrypts a bit β under a branch public key p using randomness r, producing a branch ciphertext $c = e_j(p, \beta, r)$. Let ℓ_j^r denote the number of random bits used by e_j, and let ℓ_j^{ct} denote the length of its ciphertexts. We work in the sparse ciphertext regime, where $\ell_j^{\mathsf{ct}} \geq \ell_j^r + \omega(\log \lambda)$[4].

[4]This sparsity condition is not merely a technical assumption. Dense ciphertext spaces admit simple secret-sharing concatenation- or composition-style combiners, as discussed in Section 2.4.

Thus, for every fixed public key p, only a negligible fraction of the ambient ciphertext space consists of valid ciphertexts under p.

The oracle d_j is the corresponding perfect decryption oracle: on input (s, c), it first determines the public key $p = g_j(s)$, and then checks whether c is one of the ciphertexts $e_j(p, \beta, r)$. If so it returns the corresponding bit β; otherwise it returns $\perp$.

This is the right oracle model for our separation because each branch oracle is already an ideal IND\$-CPA PKE: it has perfectly correct decryption and sparse, random-looking ciphertexts. Thus, if no black-box combiner exists even when the two branches are replaced by such ideal labelled-PKE oracles, then the failure is not due to a weakness of the underlying branches, but to the impossibility of combining them in a black-box way.

To keep track of such oracle information, we represent every revealed query-response relation as a labelled cell. A query $g_j(s)$ reveals a cell $G_{j,s} = p$. A query $e_j(p, \beta, r)$ reveals a cell $E_{j,p,\beta,r} = c$, and also forces the inverse information $\mathsf{Inv}_{j,p,c} = \beta$. Finally, a decryption query $d_j(s, c)$ is governed by the public key $p = g_j(s)$ and the inverse cell $\mathsf{Inv}_{j,p,c}$: it returns the value of this inverse cell if it is defined, and returns $\perp$ otherwise. The important point is that d_j is not an independent random table. Its answers are correlated with the g_j- and e_j-tables through these inverse cells.

In the proof, Eve's transcript will therefore be viewed as a set of revealed cells, closed under the obvious consistency rules: learning an encryption cell also reveals its inverse cell, and answering a decryption query requires knowing the relevant public-key cell and inverse cell.

Why key-generation and encryption queries are easy. Let us first suppose that the combiner decryptor only makes forward queries, i.e., queries to the g_j- and e_j-oracles, and ignore the branch decryption oracles d_0, d_1. In this restricted setting, the proof is essentially the IR argument described above. The only change is syntactic: instead of learning random-oracle query-response pairs, Eve learns labelled PKE cells.

Eve receives the combined public key pk. As in the IR attack, she runs many fresh honest combiner encryptions under pk, for both possible message bits, and records all oracle information revealed during these executions. More precisely, whenever an encryption asks a branch public-key query $g_j(s)$ and receives p, Eve records the cell $G_{j,s} = p$. Whenever an encryption asks an encryption query $e_j(p, \beta, r)$ and receives c, Eve records the cell $E_{j,p,\beta,r} = c$, and also records the forced inverse cell $\mathsf{Inv}_{j,p,c} = \beta$. Finally, a decryption query $d_j(s, c)$ is recorded through the cells that determine its answer. Let $p = g_j(s)$ be the branch public key associated with s in the oracle world, and let $a = d_j(s, c)$. Eve records the public-key cell $G_{j,s} = p$ and the inverse cell $\mathsf{Inv}_{j,p,c} = a$. Here $a \in \{0, 1, \perp\}$: if $a = \beta \in \{0, 1\}$, then c is a valid encryption of β under p, while $a = \perp$ records that c is not a valid ciphertext under p.

Let I denote the resulting closed transcript of learned cells. The role of I is exactly the same as in the ordinary IR proof: it is meant to contain all oracle information that is likely to be shared between a future challenge encryption and a key-generation view compatible with pk.

After this learning phase, Eve samples a fake combiner key-generation view $K' \leftarrow \mathsf{Samp}_{\mathsf{Keygen}}(\mathsf{pk}, I)$. This means that K' is distributed as an honest execution of $\mathsf{Keygen}^{(\cdot,\cdot)}_{\mathsf{cmb}}$, conditioned on outputting the same combined public key pk and conditioned on being consistent with all cells in I. Let sk' be the combined secret key contained in this sampled view. As before, sk' is not necessarily the real secret key. The point is only that, if the challenge encryption and K' do not disagree on any unseen oracle cell, then sk' is a valid secret key for a world in which the challenge ciphertext was honestly generated.

The good event is the direct analogue of the IR good event. Let $B^\star$ be the actual challenge-encryption view producing the challenge ciphertext $\mathsf{ct}^\star$. We want

$$\mathsf{Cells}(K') \cap \mathsf{Cells}(B^\star) \subseteq \mathsf{Cells}(I).$$

That is, every cell that appears both in the fake key-generation view and in the challenge-encryption view should already have been learned by Eve. The same heavy-query argument applies here. A cell is heavy if it appears with noticeable probability in a fresh encryption under the fixed public key pk. Since each encryption makes only polynomially many oracle queries, there can be only polynomially many heavy cells. By sampling sufficiently many encryptions, Eve learns all heavy cells with high probability. Conditioned on this, every unlearned cell appearing in K' is light for a fresh challenge encryption, and a union bound over the polynomially many cells in K' shows that the challenge encryption hits none of them except with small probability.

Now consider the final decryption using the fake key sk'. Since, in the present simplified case, the decryptor asks only g- and e-queries, Eve can answer these queries by the same mixed-world rule as in the ordinary IR proof. For a query $g_j(s)$, she answers

$$\widetilde{g}_j(s) = \begin{cases} K'(G_{j,s}), & \text{if the cell } G_{j,s} \text{ appears in } K', \\ g_j(s), & \text{otherwise, by querying the real oracle.} \end{cases}$$

Similarly, for a query $e_j(p, \beta, r)$, she answers

$$\widetilde{e}_j(p, \beta, r) = \begin{cases} K'(E_{j,p,\beta,r}), & \text{if the cell } E_{j,p,\beta,r} \text{ appears in } K', \\ e_j(p, \beta, r), & \text{otherwise, by querying the real oracle.} \end{cases}$$

When the second case occurs, Eve records the newly revealed forward cell, and in the case of an e_j-query she also records the forced inverse cell $\mathsf{Inv}_{j,p,c} = \beta$.

Forward queries are easy precisely because this mixed simulation introduces no new kind of inconsistency. If a forward query belongs to the fake key-generation view K', Eve answers according to K'. If it does not belong to K', Eve answers according to the real oracle. The only possible conflict would be a cell that is needed both by K' and by the challenge-encryption view but was not already fixed in I. This is exactly the bad event ruled out by the heavy-query argument.

Consequently, on the good event, the fake key-generation view K', the real challenge-encryption view $B^\star$, and Eve's simulated forward-oracle decryption view can all be embedded into one consistent pair of labelled-PKE oracles. In that oracle world, K' honestly generates $(\mathsf{pk}, \mathsf{sk}')$, the challenge view $B^\star$ honestly encrypts $b^\star$ to $\mathsf{ct}^\star$, and the simulated decryptor is simply the honest combiner decryptor run with sk'. Therefore correctness of the combiner implies that Eve recovers $b^\star$ with essentially the honest correctness probability, up to the small probability of the IR bad event.

Why decryption queries are dangerous. The previous discussion shows that, if the combiner decryptor used only the forward oracles g_j and e_j, then the proof would be a direct labelled-PKE version of the IR argument. The real issue is that a meaningful PKE combiner normally cannot avoid branch decryption. Even for ordinary IND-CPA PKE, the standard combiner encrypts a secret share of the message independently under each branch public keys and concatenates[5] the two

[5]Another standard way to combine ordinary PKE schemes is to encrypt the message under one branch and then encrypt the resulting ciphertext under the other branch. This variant differ syntactically from simple concatenation, but they have the same relevant feature for us. The combined decryptor must invoke the branch decryption algorithms in order to recover the message.

ciphertexts; the combined decryptor then invokes the branch decryption algorithms and combines their outputs. Similarly, in our labelled-PKE oracle model, branch ciphertexts are produced by queries of the form $c = e_j(p, \beta, r)$, but the combined decryptor does not know the randomness r. To recover the bit encrypted by c under a branch secret key s, the natural operation is therefore the branch decryption query $d_j(s, c)$. Thus, if a IND$-CPA PKE-combiner is to use the branch ciphertexts produced by its encryption algorithm in any meaningful black-box way, the proof must handle d_j-queries.

A decryption query is dangerous because it is not an independent table lookup. Its answer is derived from two pieces of oracle information. First, s determines a public key $p = g_j(s)$. Then c is interpreted using the inverse cell $\mathsf{Inv}_{j,p,c}$. If this cell is defined and equals β, then $d_j(s, c) = \beta$; otherwise $d_j(s, c) = \perp$.

This becomes subtle in Eve's mixed fake-key world. When the simulated decryptor asks $d_j(s, c)$, the public key attached to s in the simulated world is $\widetilde{p} = \widetilde{g}_j(s)$, where $\widetilde{g}_j$ is answered from K' on cells appearing in the fake key-generation view and from the real oracle otherwise. Hence $\widetilde{p}$ need not equal the real value $g_j(s)$. Therefore Eve cannot safely answer the query by asking the real oracle $d_j(s, c)$: the real oracle decrypts relative to $g_j(s)$, whereas the simulated execution must decrypt relative to $\widetilde{p}$.

For example, K' may contain $G_{j,s'} = p$. In the simulated world, s' is a secret key for p, although in the real oracle world $g_j(s')$ may be a different public key. If the simulated decryptor asks $d_j(s', c)$, the correct answer is determined by $\mathsf{Inv}_{j,p,c}$, not by the real value $d_j(s', c)$. Querying the real decryption oracle may therefore answer the wrong question.

Inverse cells and Eve's stateful simulation. The correct way to simulate branch decryption queries is to interpret them through inverse cells. During the final decryption simulation, Eve maintains a closed transcript of all cells she has learned. The transcript contains the cells learned during the encryption-sampling phase, the cells in the fake key-generation view K', and any new forward cells revealed during the simulation.

Forward queries are handled as before. If the simulated decryptor asks $g_j(s)$, Eve answers from K' when $G_{j,s}$ appears in K', and from the real g_j-oracle otherwise. If it asks $e_j(p, \beta, r)$, Eve answers from K' when $E_{j,p,\beta,r}$ appears in K', and from the real e_j-oracle otherwise. Whenever an e_j-query returns c, Eve records both $E_{j,p,\beta,r} = c$ and $\mathsf{Inv}_{j,p,c} = \beta$.

For a decryption query $d_j(s, c)$, Eve first computes the simulated public key associated with s, i.e., $\widetilde{p} = \widetilde{g}_j(s)$. She then looks at the inverse cell $\mathsf{Inv}_{j,\widetilde{p},c}$. If this inverse cell is already known, Eve returns its stored value. If it is not known, Eve returns $\perp$ and records $\mathsf{Inv}_{j,\widetilde{p},c} = \perp$. Thus Eve never answers a decryption query by blindly calling the real d_j-oracle. The real oracle would decrypt relative to the real public key $g_j(s)$, while the simulated execution must decrypt relative to the simulated public key $\widetilde{g}_j(s)$.

This simulation is stateful because later answers must remain consistent with earlier ones. Once Eve has recorded an inverse cell, subsequent queries involving the same $(j, \widetilde{p}, c)$ must receive the same answer. Similarly, if Eve later learns an encryption cell $E_{j,\widetilde{p},\beta,r} = c$, then it forces $\mathsf{Inv}_{j,\widetilde{p},c} = \beta$. The only problematic case is when this forced value contradicts an earlier recorded value $\perp$.

Bad events: unlearned intersections and false-$\perp$. The labelled-PKE simulation has two bad events. The first is the usual IR bad event: Eve may miss an intersection between the fake key-generation view and the challenge-encryption view. Formally, if $B^\star$ is the challenge-encryption

view, this bad event is

$$\mathsf{Cells}(K') \cap \mathsf{Cells}(B^\star) \not\subseteq \mathsf{Cells}(I).$$

As before, this is ruled out by the heavy-cell learning argument.

The second bad event is new and comes from decryption queries. Suppose the simulated decryptor asks $d_j(s, c)$, Eve computes $\widetilde{p} = \widetilde{g}_j(s)$, and the inverse cell $\mathsf{Inv}_{j,\widetilde{p},c}$ is not yet known. Eve then answers $\perp$. This answer is wrong exactly when c is actually a valid ciphertext under $\widetilde{p}$, namely when $c = e_j(\widetilde{p}, \beta, r)$ for some bit β and randomness r. In that case the correct inverse cell is $\mathsf{Inv}_{j,\widetilde{p},c} = \beta$, but Eve has answered $\perp$. We call this a false-$\perp$ event.

Thus, compared with the ordinary IR argument, the labelled-PKE setting has one additional source of simulation error. The usual heavy-query argument ensures that Eve does not miss the relevant key-generation/encryption intersections. The remaining task is to show that false-$\perp$ events are negligible. This is where the sparsity of the branch ciphertext spaces and the IND$-CPA security of the combiner under branch corruption enter the proof.

Sparsity plus branch-corruption security bounds false-$\perp$. It remains to explain why false-$\perp$ events are rare. Fix a branch $j \in \{0, 1\}$. Consider one decryption query during Eve's simulation, say $d_j(s, c)$. Eve first computes the simulated public key $\widetilde{p} = \widetilde{g}_j(s)$ and then looks for the inverse cell $\mathsf{Inv}_{j,\widetilde{p},c}$. If this cell is unseen, Eve answers $\perp$. This answer is incorrect only if c is in fact a valid branch ciphertext under $\widetilde{p}$, namely if $c = e_j(\widetilde{p}, \beta, r)$ for some bit β and randomness r.

For a random labelled-PKE oracle, this is a sparse event. For each fixed public key p, the number of valid ciphertexts under p is at most $2 \cdot 2^{\ell_j^r}$, one set of ciphertexts for each message bit and each encryption randomness. By contrast, the ambient branch ciphertext space has size $2^{\ell_j^{\mathsf{ct}}}$. Since $\ell_j^{\mathsf{ct}} - \ell_j^r = \omega(\log \lambda)$, this probability is negligible. Hence, on a uniformly random input, an inverse cell $\mathsf{Inv}_{j,p,c}$ associated with a fresh-looking branch ciphertext c is non-$\perp$ only with negligible probability over the initial random choice of the labelled-PKE oracle. A union bound over the polynomially many decryption queries made by the combiner decryptor shows that

$$\Pr[\mathsf{FalseBot}_j \mid \text{uniform input}] \le \mathrm{negl}(\lambda).$$

The remaining point is to transfer this bound from uniform inputs to real combiner ciphertexts. This is where the IND$-CPA security of the combiner under single-branch corruption is used. Suppose, for contradiction, that false-$\perp$ events in branch j happened with noticeably larger probability on real combiner ciphertexts than on uniformly random strings. Then a branch-j-corruption adversary could distinguish the two cases.

Indeed, after corrupting branch j, the adversary has enough information to test whether Eve's simulated $\perp$-answer was false. It can run Eve's simulation, track every branch-j decryption query $d_j(s, c)$, compute the simulated public key $\widetilde{p} = \widetilde{g}_j(s)$, and check whether the inverse cell $\mathsf{Inv}_{j,\widetilde{p},c}$ is actually non-$\perp$ in the corrupted branch oracle. Thus the event $\mathsf{FalseBot}_j$ is a testable predicate for an adversary corrupting branch j. A noticeable gap in the probability of this predicate between real combiner ciphertexts and uniform strings would directly contradict the pseudorandomness of the combiner under branch-j corruption. Consequently,

$$\Pr[\mathsf{FalseBot}_j \mid \text{real combiner ciphertext}] \le \mathrm{negl}(\lambda)$$

for each branch j. Taking a union bound over $j \in \{0, 1\}$, the total probability of any false-$\perp$ event during Eve's simulation is negligible. Thus, except with negligible probability, Eve's inverse-cell simulation answers all branch decryption queries consistently with some valid labelled-PKE oracle world.

Final contradiction. We can now combine the two parts of the analysis. The first bad event is the usual IR event: Eve may fail to learn an intersection between the fake key-generation view K' and the challenge-encryption view $B^\star$. By the heavy-cell learning argument, this happens with probability at most $\delta(\lambda)$. The second bad event is specific to the labelled-PKE setting: Eve may make a false-$\perp$ error while simulating a branch decryption query. By the sparsity and branch-corruption argument above, this happens with negligible probability.

Therefore, except with probability $\delta(\lambda) + \mathrm{negl}(\lambda)$ which is atmost $2\delta(\lambda)$ for large λ, the fake key-generation view K', the real challenge-encryption view $B^\star$, and Eve's simulated decryption view are jointly consistent. In other words, they can be embedded into a single valid pair of labelled-PKE oracles (O'_0, O'_1) such that K' honestly generates the combined key pair $(\mathsf{pk}, \mathsf{sk}')$, the challenge view honestly encrypts $b^\star$ to $\mathsf{ct}^\star$, and Eve's final simulation is exactly the combiner decryptor run on $(\mathsf{sk}', \mathsf{ct}^\star)$ in this same oracle world.

By correctness of the combiner, decryption in such a consistent oracle world outputs the encrypted bit with probability equal to the honest correctness probability, say $\rho(\lambda)$. Hence, on a real challenge ciphertext, Eve recovers the hidden bit with probability at least $\rho(\lambda) - 2\delta(\lambda)$.

On the other hand, if the challenge string is uniformly random, then it is independent of the hidden bit $b^\star$. Eve's entire view in this experiment, including the public key, her learned transcript, the fake key-generation view, and the simulated decryption transcript, is independent of $b^\star$. Thus her output can predict $b^\star$ with probability exactly $\frac{1}{2}$.

Therefore Eve distinguishes real combiner ciphertexts from uniformly random strings with advantage at least $\rho(\lambda) - \frac{1}{2} - 2\delta(\lambda)$. If the combiner has overwhelming correctness, then $\rho(\lambda) = 1 - \mathrm{negl}(\lambda)$, and the above advantage is non-negligible. This contradicts the assumed IND$-CPA security of the combiner.

Hence no polynomial-query black-box $(2, 1)$-combiner for IND$-CPA PKE exists in the random labelled-PKE oracle model, even when the combiner decryptor is allowed to query the full branch oracles.

Why the sparse ciphertext regime is necessary. The sparse ciphertext assumption is not merely a technical convenience; it is the property that makes the lower-bound proof go through. Recall that, for a fixed branch public key p, the number of valid ciphertexts is at most $2 \cdot 2^{\ell_j^r}$, whereas the ambient ciphertext space has size $2^{\ell_j^{\mathsf{ct}}}$. Thus the density of valid ciphertexts under p is at most $2^{1-(\ell_j^{\mathsf{ct}} - \ell_j^r)}$. When $\ell_j^{\mathsf{ct}} - \ell_j^r = \omega(\log \lambda)$, this density is negligible. Consequently, a uniformly random string is almost never a valid ciphertext under a fixed public key.

This is exactly the property used in the proof. During Eve's inverse-cell simulation, when she encounters an unseen inverse cell $\mathsf{Inv}_{j,p,c}$, she answers $\perp$. In the sparse regime, this is correct with overwhelming probability on a uniformly random input, because a fresh-looking ciphertext string c is very unlikely to lie in the small set of valid encryptions under p. The proof then uses the pseudorandomness of the combiner under branch corruption to transfer this negligible false-$\perp$ probability from uniform inputs to real combiner ciphertexts.

If the ciphertext space were dense, this argument would fail. For example, if $\ell_j^{\mathsf{ct}} = \ell_j^r + 1$ and, for every public key p, the encryption map were a permutation, then every ambient ciphertext string would be valid under p. A uniformly random ciphertext would no longer decrypt to $\perp$; instead, it would decrypt to a plausible uniformly random bit. In such a model, Eve's default $\perp$-answer to unseen inverse cells would be wrong with large probability, and the false-$\perp$ event would no longer be negligible.

Thus the sparse regime is precisely where the inverse-cell simulation is valid. In the dense regime, random ciphertext strings can look like honest ciphertexts, and the standard combining ideas discussed in Section 2.3 may apply. Therefore our impossibility result should be read as a lower bound for black-box $(2,1)$-combiners for IND$-CPA PKE in the sparse ciphertext regime, not as a statement ruling out dense-ciphertext combining strategies.

3 Preliminaries

Notations: Let PPT denote probabilistic polynomial-time. We denote the set of all positive integers up to n as $[n] := \{1, \ldots, n\}$. For any two binary strings x, y, we use the notation $x \| y$ to denote x concatenated with y. Here, x_i denotes the ith bit of x. We denote $\mathsf{Ham}(x)$ as the Hamming weight of x, i.e., the number of '1' in the binary string x. Similarly, $\mathsf{Ham}(x, y)$ denotes the Hamming distance between the binary strings x and y, i.e., the number of indices where x and y differ. We denote $\mathcal{B}_r^n(x)$ as the set $\{x' \in \{0,1\}^n \mid \mathsf{Ham}(x, x') \leq r\}$, i.e., the set of all strings with Hamming distance at most r from x. For any finite set S, $x \leftarrow S$ denotes a uniformly random element x from the set S. Similarly, for any distribution $\mathcal{D}$, $x \leftarrow \mathcal{D}$ denotes an element x drawn from distribution $\mathcal{D}$. We use λ to denote the security parameter.

3.1 Probability and Linear Algebra over the Reals

Definition 3.1 (Bernoulli distribution)**.** *A binary random variable $x \in \{0,1\}$ is said to follow a **Bernoulli distribution** with bias parameter $\tau \in [0,1]$, denoted as Ber_τ, if $\Pr[x = 1] = \frac{1-\tau}{2}$. We say that $\mathbf{x} \in \{0,1\}^n$ follows Ber_τ^n, if each component x_i of $\mathbf{x}$ independently follows Ber_τ.*

Lemma 3.2 (Chernoff-Hoeffding Bound)**.** *Let $X_1, .., X_k$ be independent random variables with $\forall i : \Pr[X_i = 1] = p, \Pr[X_i = 0] = 1 - p$ where $0 < p < 1$. Now, for $\epsilon \in [0,1]$:*

$$\Pr\left[\sum_{i=1}^{k} X_i \leq (1-\epsilon)kp\right] \leq 2^{-kp\epsilon^2/2}$$

$$\Pr\left[\sum_{i=1}^{k} X_i \geq (1+\epsilon)kp\right] \leq 2^{-kp\epsilon^2/3}$$

3.2 Secret-key 0-bit Pseudorandom Codes

Definition 3.3 (Secret-key 0-bit PRC)**.** *A secret-key PRC with κ-correctness and τ-robustness is described by two algorithms $(\mathsf{Enc}, \mathsf{Dec})$[6] parameterized by a secret key sk, satisfying the following criteria:*

- ***(Syntax)** There exist functions $s, n, r, r' \in \mathbb{N} \to \mathbb{N}$ such that for all $\lambda \in \mathbb{N}$, $\mathsf{Enc} : \{0,1\}^{s(\lambda)} \times \{0,1\}^{r'(\lambda)} \to \{0,1\}^{n(\lambda)}$, and $\mathsf{Dec} : \{0,1\}^{s(\lambda)} \times \{0,1\}^{n(\lambda)} \times \{0,1\}^{r(\lambda)} \to \{\top, \bot\}$.*

[6]We deliberately omit the Keygen algorithm, since in the secret-key setting the secret key can be taken to be a uniformly random string.

- ***(Error correction, or robustness)*** *For any* $\lambda \in \mathbb{N}$*,*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, c+e) \neq \top \ : \ \begin{array}{r} \mathsf{sk} \leftarrow \{0,1\}^{s(\lambda)} \\ c \leftarrow \mathsf{Enc}(\mathsf{sk};\rho) \\ e \leftarrow \mathsf{Ber}_\tau^n \end{array}\right] \ \leq \ \kappa(\lambda).$$

where the probability is over the randomness of $\mathsf{Enc}, \mathsf{Dec}$ *and error* e*.*

- ***(Pseudorandomness)*** *For any polynomial-time adversary* $\mathcal{A}$*,*

$$\left| \Pr_{\mathsf{sk}\leftarrow\{0,1\}^{s(\lambda)}}\left[\mathcal{A}^{\mathsf{O}_{\mathsf{sk}}}(1^\lambda)=1\right] \ - \ \Pr_{U_\lambda}\left[\mathcal{A}^{U_\lambda}(1^\lambda)=1\right] \right| \ \leq \ \mathrm{negl}(\lambda),$$

where O_{sk} *is an inputless oracle that samples a random* $\rho \in \{0,1\}^{r'(\lambda)}$ *and outputs* $\mathsf{Enc}(\mathsf{sk};\rho)$ *and* U_λ *denotes an oracle that returns a fresh uniform string in* $\{0,1\}^n$*.*

3.3 Public-Key Encryption

A public-key encryption scheme (PKE) for single bit message[7] is a triple of PPT algorithms $\mathsf{PKE} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$:

$\mathsf{Keygen}(1^\lambda)$**:** The setup algorithm takes as input a unary representation of the security parameter λ and outputs a public key pk and secret key sk.

$\mathsf{Enc}(\mathsf{pk}, m)$**:** The encryption algorithm takes as input a public key pk and a message $m \in \{0,1\}$ and outputs a ciphertext $\mathsf{ct} \in \{0,1\}^{\ell_{\mathsf{ct}}}$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$**:** The decryption algorithm takes as input a secret key sk and a ciphertext ct and outputs either a message $m \in \{0,1\}$ or $\perp$.

A scheme PKE is *correct* if there exists a function $\epsilon(\lambda) = \mathrm{negl}(\lambda)$ such that for all $\lambda \in \mathbb{N}, m \in \{0,1\}$ and pk, sk in the support of Keygen,

$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) = m] \geq 1 - \epsilon(\lambda),$$

where the probability is taken over the randomness of Enc and Dec. We say that PKE is *perfectly* correct if $\epsilon(\lambda) = 0$.

Definition 3.4 ([RBBK01])**.** *A PKE scheme is said to be* ***IND$-CPA*** *secure if for all stateful PPT adversaries* Adv*, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$*,*

$$\Pr\left[b' = b \ : \ \begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Keygen}(1^\lambda) \\ m \leftarrow \mathsf{Adv}(\mathsf{pk}), \mathsf{ct}_0 \leftarrow \{0,1\}^{\ell_{\mathsf{ct}}} \\ \mathsf{ct}_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m) \\ b \leftarrow \{0,1\}, b' \leftarrow \mathsf{Adv}(\mathsf{ct}_b) \end{array}\right] \leq \frac{1}{2} + \mathrm{negl}(\lambda).$$

where the probability is over the randomness of $\mathsf{Keygen}, \mathsf{Enc}$ *and* Adv*.*

[7] In fact, any single-bit PKE scheme can be extended to encrypt multi-bit messages by encrypting each bit independently, in a black-box manner.

4 Impossibility of Combiners for Pseudorandom Codes

In this and the following sections, we will show our first main result of the paper, the impossibility of constructing a black-box combiner for pseudorandom codes.

An (n,n-k)-robust black-box combiner for PRCs takes n implementations of PRCs, and builds a new PRC that is secure if at least n-k of the n implementations are secure. More formally, we allow the adversary to adaptively corrupt up to k implementations, and for each corrupted implementation, the adversary gains access to the truth table of the implementation.

Definition 4.1 ((n,n-k)-robust PRC Combiner)**.** *Let us fix a security parameter λ. Let $\mathsf{PRC}_1, \ldots, \mathsf{PRC}_n$ with $\mathsf{PRC}_j = (\mathsf{Enc}_j, \mathsf{Dec}_j)$ denote n PRC implementations, with secret key length k_i, randomness length ρ_i and codeword length n_i. Further, let κ_j be the correctness error and τ_j be the decoding radius. An $(n, n-k)$ PRC-combiner consists of two efficient randomized oracle algorithms $\mathsf{PRC}_{\mathsf{cmb}}^{(\cdot)} = (\mathsf{Enc}_{\mathsf{cmb}}^{(\cdot)}, \mathsf{Dec}_{\mathsf{cmb}}^{(\cdot)})$, where both algorithms have oracle access to the component oracle pairs $\mathsf{PRC}_1, \ldots, \mathsf{PRC}_n$ with $\mathsf{PRC}_j = (\mathsf{Enc}_j, \mathsf{Dec}_j)$. The combined algorithms have the syntax*

$$\mathsf{Enc}_{\mathsf{cmb}}^{\mathsf{PRC}_1,\ldots,\mathsf{PRC}_n} : \{0,1\}^{k_{\mathsf{cmb}}} \times \{0,1\}^{\rho_{\mathsf{cmb}}} \to \{0,1\}^{n_{\mathsf{cmb}}},$$

and

$$\mathsf{Dec}_{\mathsf{cmb}}^{\mathsf{PRC}_1,\ldots,\mathsf{PRC}_n} : \{0,1\}^{k_{\mathsf{cmb}}} \times \{0,1\}^{n_{\mathsf{cmb}}} \to \{\top, \bot\}.$$

Further, we define the correctness error $\kappa_{\mathsf{cmb}} := \kappa_{\mathsf{cmb}}(\kappa_1, \ldots, \kappa_n)$ and decoding radius $\tau_{\mathsf{cmb}} := \tau_{\mathsf{cmb}}(\tau_1, \ldots, \tau_n)$. We require the following two properties.

1. ***Correctness and robustness.*** *For every collection of component oracle pairs $\mathsf{PRC}_1, \ldots, \mathsf{PRC}_n$ such that each PRC_j is a κ_j-correct and τ_j-robust PRC with output length n_j, the combined PRC*

$$\mathsf{PRC}_{\mathsf{cmb}}^{\mathsf{PRC}_1,\ldots,\mathsf{PRC}_n} = \left(\mathsf{Enc}_{\mathsf{cmb}}^{\mathsf{PRC}_1,\ldots,\mathsf{PRC}_n}, \mathsf{Dec}_{\mathsf{cmb}}^{\mathsf{PRC}_1,\ldots,\mathsf{PRC}_n}\right)$$

is a κ_{cmb}-correct and τ_{cmb}-robust PRC with output length N_{cmb} for some fixed functions $\kappa_{\mathsf{cmb}}, \tau_{\mathsf{cmb}}, N_{\mathsf{cmb}}$. The probability in the correctness guarantee is taken over the internal randomness of the combiner algorithms.

2. ***Security.*** *Security is defined by the following experiment with an unbounded adversary $\mathcal{A}$ making polynomially many oracle queries.*

   (a) *The adversary $\mathcal{A}$ is given the public description of the combiner and all parameters, as well as oracle access to all PRC_j.*

   (b) *The experiment then enters a corruption phase. During this phase, the adversary may query a $\mathsf{Corrupt}(i)$ oracle. On such a query, the experiment adds i to the corrupted set S and returns the full truth-table description of the i-th PRC, namely $\mathsf{PRC}_i = (\mathsf{Enc}_i, \mathsf{Dec}_i)$.*

   (c) *Once the adversary ends the corruption phase, the set S is fixed. From this point on, no further corruption queries are allowed. The challenger samples a random bit b and a combined secret key $\mathsf{sk}_{\mathsf{cmb}} \leftarrow \{0,1\}^{k_{\mathsf{cmb}}}$. The adversary then obtains access to the challenge oracle $\mathsf{Challenge}(1^\lambda)$, which behaves as follows:*

      i. *If $b = 1$, it samples $\rho \leftarrow \{0,1\}^{\rho_{\mathsf{cmb}}}$ uniformly and returns $\mathsf{Enc}_{\mathsf{cmb}}^{\mathsf{PRC}_1,\ldots,\mathsf{PRC}_n}(\mathsf{sk}_{\mathsf{cmb}}, \rho)$.*

      ii. *If $b = 0$, it returns an independent uniform string $U_{n_{\mathsf{cmb}}} \leftarrow \{0,1\}^{n_{\mathsf{cmb}}}$.*

(d) Finally, the adversary outputs a bit b'.

The adversary wins the game if $b = b'$ and $|S| \leq k$. The advantage of $\mathcal{A}$ in the above experiment is

$$\mathsf{Adv}_{\mathcal{A}} := \left| \Pr[b' = b] - \frac{1}{2} \right|,$$

where the probability is taken over any randomness of the underlying PRC_js, the challenge key $\mathsf{sk}_{\mathsf{cmb}}$, the fresh randomness used by the challenge oracle and the internal randomness of the combiner algorithms.

For the combiner to be secure in the random PRC model, we require that for every unbounded adversary $\mathcal{A}$ making polynomially many oracle queries, there exists a negligible function $\mathrm{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $\mathsf{Adv}_{\mathcal{A}} \leq \mathrm{negl}(\lambda)$.

We will now define the branch implementations used in the separation. We use lazily-sampled random PRCs, which can be viewed as PRCs generated from a hidden random codebook; Appendix B explains the closely related separated-codebook viewpoint used in the security proof. Note that for a black-box combiner, the underlying implementations do not necessarily need to be efficiently implemented.

Definition 4.2 ((Lazy-sampled) Random PRCs)**.** *Fix any $j \in [n]$ and any constant $\tau \in (0, 1/4)$. Let $\mathsf{PRC}_j = (\mathsf{Enc}_j, \mathsf{Dec}_j)$ be defined over secret-key space $\{0,1\}^{k_j}$, randomness space $\{0,1\}^{r_j}$, and codeword space $\{0,1\}^{n_j}$, where $k = \lambda, r = \lambda, n = \lambda^2$.*

For every pair $(\kappa, r) \in \{0,1\}^{k_j} \times \{0,1\}^{r_j}$, sample independently $C^{(j)}_{\kappa,r} \leftarrow \{0,1\}^{n_j}$. Define:

- $\mathsf{Enc}_j(\kappa; r) := C^{(j)}_{\kappa,r}$,
- $\mathsf{Dec}_j(\kappa, x) := \top$ *if there exists $r \in \{0,1\}^{r_j}$ such that $\mathsf{Ham}(x, C^{(j)}_{\kappa,r}) \leq \tau n_j$. Otherwise $\mathsf{Dec}_j(\kappa, x) := \bot$.*

Lemma 4.3. *PRC_j is a pseudorandom code (with negligible correctness error in the Bernoulli error model).*

We defer the proof of Lemma 4.3 to Appendix B. This allows us now to state our main theorem for this section, the impossibility of a black-box combiner for pseudorandom codes:

Theorem 4.4 (No Combiner for PRCs)**.** *There exists no (n,n-1)-robust black-box combiner for pseudorandom codes with correctness error at most $\frac{1}{16}$.*

Note that a (n,n-1)-robust combiner is the weakest variant of a combiner, where at most one out of n implementations are allowed to be broken. Therefore, our result extends to any (n,n-k)-robust combiner.

The proof of Theorem 4.4 is split over the next three sections. The main idea is that we can compile out the oracle queries of $\mathsf{Dec}_{\mathsf{cmb}}$, showing in the end that there exists a nearly equivalent decoder without any oracle queries, which leads to a contradiction because there does not exist multi-time information-theoretic secure PRCs. In the next two sections, we will show that we can compile out the first query if it is a Dec_i or Enc_i query, respectively. Afterwards, we will combine both results and show how to compile out all queries iteratively.

5 Compiling out Decode queries

In this section, we show that the first oracle query made by the combiner decoder $\mathsf{Dec}_{\mathsf{cmb}}$ is essentially uninformative if it is a decryption query. At a high level, our argument has two steps. First, we prove a transfer statement: if the decoding of the first query produced from an honestly generated combiner codeword were noticeably different from the decoding of the first query produced from a uniformly random string, then this gap could be converted into a distinguisher against the pseudorandomness of the combiner itself. Hence, pseudorandomness of the combiner implies that, for almost all keys, the first-query behavior on a real codeword is statistically close to the first-query behavior on a uniform string (assuming that the corresponding underlying PRC can be corrupted).

The second step is to analyze what happens when this first query is issued to our random lazy-sampled PRC. Note that because our combiner is supposed to be black-box, it should also work if the underlying PRCs are implemented inefficiently. We show that, with overwhelming probability over the choice of the random PRC oracle, the set of strings accepted by the branch decoder is extremely sparse. Consequently, when the first query is evaluated on a uniformly random input, the decoder returns $\perp$ with overwhelming probability. Combining this sparse-acceptance property with the transfer statement from the first step, we conclude that the same is true when the input to $\mathsf{Dec}_{\mathsf{cmb}}$ is an honestly generated combiner codeword: with high probability, the first branch query also receives the answer $\perp$. In this sense, the first query made by $\mathsf{Dec}_{\mathsf{cmb}}$ is useless on a random PRC oracle.

For this, let $\mathsf{PRC}_{\mathsf{cmb}} = (\mathsf{Enc}_{\mathsf{cmb}}, \mathsf{Dec}_{\mathsf{cmb}})$ be a black-box combiner for zero-bit secret-key pseudorandom codes $\mathsf{PRC}_1, \ldots, \mathsf{PRC}_n$ with secret key length k_{cmb} and codeword length n_{cmb}. For every key $\mathsf{sk} \in \{0,1\}^{k_{\mathsf{cmb}}}$ and every input $x \in \{0,1\}^{n_{\mathsf{cmb}}}$, we define by

$$\mathsf{Qry}_{\mathsf{sk}}(x; \rho_{\mathsf{Qry}}) = (j, \kappa, cw_0),$$

the first query made by $\mathsf{Dec}_{\mathsf{cmb}}(\mathsf{sk}, x)$, where ρ_{Qry} denotes the internal randomness of Qry, and $\mathsf{Dec}_{\mathsf{cmb}}$ then queries $\mathsf{Dec}_j(\kappa, cw_0)$. Note that $\mathsf{Qry}_{\mathsf{sk}}$ is efficiently computable from the description of $\mathsf{Dec}_{\mathsf{cmb}}$.

Theorem 5.1. *Let $\mathsf{PRC}_{\mathsf{cmb}}$ be a secure $(n, n-1)$-robust combiner for PRCs, and let $(\mathsf{PRC}_1, \ldots, \mathsf{PRC}_n)$ be sampled independently according to Definition 4.2. Then, for any inverse polynomial ε, with probability $1 - 2^{-\lambda+n}$ over the sampled branch PRCs, for a $1 - \varepsilon(\lambda)$ fraction of keys,*

$$\Pr_{\substack{y \leftarrow \mathsf{Enc}_{\mathsf{cmb}}(\mathsf{sk}) \\ \rho_{\mathsf{Qry}}}} \left[\mathsf{Dec}_j(\kappa, cw_0) = \perp \mid (j, \kappa, cw_0) = \mathsf{Qry}_{\mathsf{sk}}(y; \rho_{\mathsf{Qry}}) \right] \;\geq\; 1 - \varepsilon(\lambda).$$

Note that we don't necessarily know which Dec_j $\mathsf{Qry}_{\mathsf{sk}}$ will query first, and we can only corrupt one of the underlying PRCs. Therefore, we will consider the choice of j as n different branches. For these branches $j \in [n]$ and every sk, we now define the probabilities that the decoding of the first query is $\perp$:

$$p^{\mathrm{real}}_{\mathsf{sk},j} := \Pr_{\substack{y \leftarrow \mathsf{Enc}_{\mathsf{cmb}}(\mathsf{sk}) \\ \rho_{\mathsf{Qry}}}} \left[j' = j \;\wedge\; \mathsf{Dec}_j(\kappa, cw_0) = \perp \;\middle|\; (j', \kappa, cw_0) = \mathsf{Qry}_{\mathsf{sk}}(y; \rho_{\mathsf{Qry}}) \right]$$

$$p^{\mathrm{rand}}_{\mathsf{sk},j} := \Pr_{\substack{u \leftarrow \{0,1\}^{n_{\mathsf{cmb}}} \\ \rho_{\mathsf{Qry}}}} \left[j' = j \;\wedge\; \mathsf{Dec}_j(\kappa, cw_0) = \perp \;\middle|\; (j', \kappa, cw_0) = \mathsf{Qry}_{\mathsf{sk}}(u; \rho_{\mathsf{Qry}}) \right].$$

Further, we define similar probabilities for the case where the decoder returns $\top$:

$$a^{\text{real}}_{\mathsf{sk},j} := \Pr_{\substack{y \leftarrow \mathsf{Enc}_{\mathsf{cmb}}(\mathsf{sk}) \\ \rho_{\mathsf{Qry}}}} \Big[j' = j \ \wedge \ \mathsf{Dec}_j(\kappa, cw_0) \neq \bot \ \Big| \ (j', \kappa, cw_0) = \mathsf{Qry}_{\mathsf{sk}}(y; \rho_{\mathsf{Qry}}) \Big].$$

$$a^{\text{rand}}_{\mathsf{sk},j} := \Pr_{\substack{u \leftarrow \{0,1\}^{n_{\mathsf{cmb}}} \\ \rho_{\mathsf{Qry}}}} \Big[j' = j \ \wedge \ \mathsf{Dec}_j(\kappa, cw_0) \neq \bot \ \Big| \ (j', \kappa, cw_0) = \mathsf{Qry}_{\mathsf{sk}}(u; \rho_{\mathsf{Qry}}) \Big].$$

5.1 Gap implies break

We will now show the first part, namely that a noticeable gap between the first query for an honestly generated combiner codeword and the first query for a uniformly random string leads to a distinguisher against the pseudorandomness of the combined PRC.

Lemma 5.2. *Fix $j \in [n]$. Suppose there exist inverse-polynomial functions $\alpha(\lambda), \beta(\lambda) > 0$ such that*

$$\Pr_{\mathsf{sk} \leftarrow \{0,1\}^{k_{\mathsf{cmb}}}} \Big[\big| p^{\text{real}}_{\mathsf{sk},j} - p^{\text{rand}}_{\mathsf{sk},j} \big| \geq \beta(\lambda) \Big] \ \geq \ \alpha(\lambda).$$

Then there exists an unbounded adversary $\mathcal{A}_j$, that corrupts PRC_j (and hence obtains the complete truth table of Dec_j) and breaks the pseudorandomness of $\mathsf{PRC}_{\mathsf{cmb}}$ with inverse-polynomial advantage.

Remark 5.3. *Lemma 5.2 is completely black-box with respect to the corrupted branch PRC_j. Its proof uses only the fact that the adversary, upon corrupting PRC_j, obtains the full truth table of Dec_j. In particular, the theorem does not rely on any specific instantiation or distributional property of the branch PRC.*

Proof. We construct an adversary $\mathcal{A}_j$ in the pseudorandomness game for $\mathsf{PRC}_{\mathsf{cmb}}$. Recall that $\mathcal{A}_j$ is given oracle access to a challenge oracle F, which is either:

- the *real* oracle $F = \mathsf{Enc}_{\mathsf{cmb}}(\mathsf{sk}^\star)$ for a secret key $\mathsf{sk}^\star \leftarrow \{0,1\}^{k_{\mathsf{cmb}}}$, where each invocation uses fresh encoder randomness, or
- the *random* oracle $F = U_{n_{\mathsf{cmb}}}$, which on every query returns an independent uniform string in $\{0,1\}^{n_{\mathsf{cmb}}}$.

Now, if we would know the secret key $\mathsf{sk}^\star$, the adversary would be trivial: We could just compute $\mathsf{Qry}_{\mathsf{sk}^\star}$ and guess based on $p^{\text{real}}_{\mathsf{sk}^\star,j}$ and $p^{\text{rand}}_{\mathsf{sk}^\star,j}$ whether we are more likely to be in the real or the random world. Now, we don't know $\mathsf{sk}^\star$, but we can use that our adversary is computationally unbounded and we access to the truth table of Dec_j by corrupting PRC_j. We will generate a large number of samples x_i from F and then check for every secret key how often it decodes to $\bot$. If we are in the random world, for every key, it should be close to p^{rand}_j (as it is independent of the secret key sk). However, in the real world, there should be one $\mathsf{sk} = \mathsf{sk}^*$ for which the probability should be close to $p^{\text{real}}_{\mathsf{sk},j}$. We will now describe the adversary in detail:

The adversary. Fix $N = 4 \cdot \frac{k_{\mathsf{cmb}}+\lambda}{\beta(\lambda)^2}$. The adversary $\mathcal{A}_j$ proceeds as follows:

1. Query N times the challenge oracle F and obtains samples $x_1, \ldots, x_N \in \{0,1\}^{n_{\mathsf{cmb}}}$.
2. Sample independent coins $\rho_1, \ldots, \rho_N$ for the randomized procedure Qry.

3. For every candidate key $\mathsf{sk} \in \{0,1\}^{k_{\mathsf{cmb}}}$, compute

$$Z_{\mathsf{sk},j} \;:=\; \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\big[j_i' = j \;\wedge\; \mathsf{Dec}_j(\kappa_i, cw_i) = \bot\big]\,,$$

where $(j_i', \kappa_i, cw_i) = \mathsf{Qry}_{\mathsf{sk}}(x_i; \rho_i)$.

4. Output 1 (meaning "real") iff there exists a candidate key sk such that

$$\big|Z_{\mathsf{sk},j} - p^{\mathrm{rand}}_{\mathsf{sk},j}\big| \;\geq\; \frac{\beta(\lambda)}{2}.$$

Otherwise, output 0.

We now analyze the success probability of $\mathcal{A}_j$ in the two worlds.

Random world. Assume first that $F = U_{n_{\mathsf{cmb}}}$, so that $x_1, \ldots, x_N$ are independent and uniformly distributed in $\{0,1\}^{n_{\mathsf{cmb}}}$. Fix any candidate key $\mathsf{sk} \in \{0,1\}^{k_{\mathsf{cmb}}}$. Then the random variables

$$Y_i^{(\mathsf{sk},j)} \;:=\; \mathbf{1}\big[j_i' = j \;\wedge\; \mathsf{Dec}_j(\kappa_i, cw_i) = \bot\big]\,,$$

where $(j_i', \kappa_i, cw_i) = \mathsf{Qry}_{\mathsf{sk}}(x_i; \rho_i)$, are independent Bernoulli random variables with mean exactly $p^{\mathrm{rand}}_{\mathsf{sk},j}$. Hence

$$Z_{\mathsf{sk},j} = \frac{1}{N}\sum_{i=1}^{N} Y_i^{(\mathsf{sk},j)} \qquad \text{and} \qquad \mathbb{E}[Z_{\mathsf{sk},j}] = p^{\mathrm{rand}}_{\mathsf{sk},j}.$$

By Hoeffding's inequality,

$$\Pr\left[\big|Z_{\mathsf{sk},j} - p^{\mathrm{rand}}_{\mathsf{sk},j}\big| \geq \frac{\beta(\lambda)}{2}\right] \;\leq\; 2\exp\left(-\frac{N\beta(\lambda)^2}{2}\right).$$

There are $2^{k_{\mathsf{cmb}}}$ candidate keys. Therefore, by a union bound,

$$\Pr\left[\exists \mathsf{sk} \in \{0,1\}^{k_{\mathsf{cmb}}} : \big|Z_{\mathsf{sk},j} - p^{\mathrm{rand}}_{\mathsf{sk},j}\big| \geq \frac{\beta(\lambda)}{2}\right] \;\leq\; 2^{k_{\mathsf{cmb}}+1}\exp\left(-\frac{N\beta(\lambda)^2}{2}\right).$$

By our choice of N, the right-hand side is $2^{-\lambda}$. Thus, in the random world, $\Pr[\mathcal{A}_j^{U_{n_{\mathsf{cmb}}}}(1^\lambda) = 1] \leq \mathrm{negl}(\lambda)$.

Real world. Now assume that $F = \mathsf{Enc}_{\mathsf{cmb}}(\mathsf{sk}^\star)$ for a hidden key $\mathsf{sk}^\star \leftarrow \{0,1\}^{k_{\mathsf{cmb}}}$. Let G_j denote the set of secret keys for which there is a noticeable difference between the real and the random world:

$$G_j \;:=\; \Big\{\mathsf{sk} \in \{0,1\}^{k_{\mathsf{cmb}}} : \big|p^{\mathrm{real}}_{\mathsf{sk},j} - p^{\mathrm{rand}}_{\mathsf{sk},j}\big| \geq \beta(\lambda)\Big\}.$$

By hypothesis,

$$\Pr_{\mathsf{sk}^\star \leftarrow \{0,1\}^{k_{\mathsf{cmb}}}}[\mathsf{sk}^\star \in G_j] \;\geq\; \alpha(\lambda).$$

Condition on the event $\mathsf{sk}^\star \in G_j$. Consider the specific candidate key $\mathsf{sk} = \mathsf{sk}^\star$ used by the adversary in Step 3. Since each challenge sample x_i is an independent output of $\mathsf{Enc}_{\mathsf{cmb}}(\mathsf{sk}^\star)$, and since $\rho_1, \ldots, \rho_N$ are independent fresh coins for Qry, the random variables

$$Y_i^{(\mathsf{sk}^\star,j)} \;:=\; \mathbf{1}[b_i = j \;\wedge\; \mathsf{Dec}_j(\kappa_i, cw_i) = \bot]\,, \qquad \text{where } (b_i, \kappa_i, cw_i) = \mathsf{Qry}_{\mathsf{sk}^\star}(x_i; \rho_i),$$

are independent Bernoulli random variables with mean exactly $p^{\mathrm{real}}_{\mathsf{sk}^\star,j}$. Hence $\mathbb{E}[Z_{\mathsf{sk}^\star,j}] = p^{\mathrm{real}}_{\mathsf{sk}^\star,j}$. Again by Hoeffding's inequality,

$$\Pr\left[\left|Z_{\mathsf{sk}^\star,j} - p^{\mathrm{real}}_{\mathsf{sk}^\star,j}\right| \geq \frac{\beta(\lambda)}{2} \;\middle|\; \mathsf{sk}^\star \in G_j\right] \;\leq\; 2\exp\left(-\frac{N\beta(\lambda)^2}{2}\right) \;\leq\; \mathrm{negl}(\lambda).$$

Since $\mathsf{sk}^\star \in G_j$, we have $\left|p^{\mathrm{real}}_{\mathsf{sk}^\star,j} - p^{\mathrm{rand}}_{\mathsf{sk}^\star,j}\right| \geq \beta(\lambda)$. Therefore, except with negligible probability, $\left|Z_{\mathsf{sk}^\star,j} - p^{\mathrm{rand}}_{\mathsf{sk}^\star,j}\right| \geq \frac{\beta(\lambda)}{2}$, which implies that $\mathcal{A}_j$ accepts. Consequently,

$$\Pr[\mathcal{A}_j^{\mathsf{Enc}_{\mathsf{cmb}}(\mathsf{sk}^\star)}(1^\lambda) = 1 \mid \mathsf{sk}^\star \in G_j] \;\geq\; 1 - \mathrm{negl}(\lambda).$$

Averaging over the choice of $\mathsf{sk}^\star$, we obtain

$$\Pr[\mathcal{A}_j^{\mathsf{Enc}_{\mathsf{cmb}}(\mathsf{sk}^\star)}(1^\lambda) = 1] \;\geq\; \Pr[\mathsf{sk}^\star \in G_j] \cdot (1 - \mathrm{negl}(\lambda)) \;\geq\; \alpha(\lambda) - \mathrm{negl}(\lambda).$$

Advantage. Combining the two cases,

$$\mathsf{Adv}_{\mathcal{A}_j}(\lambda) \;\geq\; \big(\alpha(\lambda) - \mathrm{negl}(\lambda)\big) - \mathrm{negl}(\lambda) \;=\; \alpha(\lambda) - \mathrm{negl}(\lambda),$$

which is inverse-polynomial in λ. Hence $\mathcal{A}_j$ breaks the pseudorandomness of $\mathsf{PRC}_{\mathsf{cmb}}$ with inverse-polynomial advantage. This concludes the proof. □

Corollary 5.4. *Assume that $\mathsf{PRC}_{\mathsf{cmb}}$ is pseudorandom against adversaries that may corrupt PRC_j, for each $j \in [n]$. Then for every inverse-polynomial function $\beta(\lambda)$ and every $j \in [n]$,*

$$\Pr_{\mathsf{sk}\leftarrow\{0,1\}^{k_{\mathsf{cmb}}}}\left[\left|p^{\mathrm{real}}_{\mathsf{sk},j} - p^{\mathrm{rand}}_{\mathsf{sk},j}\right| \geq \beta(\lambda)\right]$$

is negligible in λ.

5.2 PRC Oracle on Uniform Input

In this section, we show that for a random lazy-sampled PRC oracle as defined in Definition 4.2, the response of the oracle on the first query evaluated on a uniformly random string is $\perp$ with overwhelming probability.

Lemma 5.5. *Let*

$$\mu_\lambda \;:=\; 2^{r_j} \cdot \frac{\mathsf{Vol}_{n_j}(\tau n_j)}{2^{n_j}} \;=\; 2^{-\Omega(\lambda^2)}.$$

Then for every function $\eta(\lambda) \in (0,1]$, with probability at least $1 - \eta(\lambda)$ over the lazy sampling of PRC_j, the following holds: for every function $t(\lambda) > 0$,

$$\Pr_{\mathsf{sk}\leftarrow\{0,1\}^{k_{\mathsf{cmb}}}}\left[a^{\mathrm{rand}}_{\mathsf{sk},j} > t(\lambda)\right] \;\leq\; \frac{\mu_\lambda}{\eta(\lambda)\, t(\lambda)}.$$

Proof. For a fixed $\kappa \in \{0,1\}^{k_j}$, the accepting region of $\mathsf{Dec}_j(\kappa,\cdot)$ is $A^{(j)}_\kappa := \bigcup_{r\in\{0,1\}^{r_j}} B_{n_j}(C^{(j)}_{\kappa,r}, \tau n_j)$, where $B_{n_j}(c, \tau n_j)$ is the Hamming ball of radius τn_j around c. By the union bound,

$$\frac{|A^{(j)}_\kappa|}{2^{n_j}} \;\leq\; \sum_{r\in\{0,1\}^{r_j}} \frac{|B_{n_j}(C^{(j)}_{\kappa,r}, \tau n_j)|}{2^{n_j}} \;=\; 2^{r_j} \cdot \frac{\mathsf{Vol}_{n_j}(\tau n_j)}{2^{n_j}} \;=\; \mu_\lambda. \tag{1}$$

Using the standard bound $\mathsf{Vol}_{n_j}(\tau n_j) \le 2^{H(\tau)n_j}$, where $H(\tau)$ denotes the binary entropy function, we obtain $\mu_\lambda \le 2^{r_j-(1-H(\tau))n_j} = 2^{\lambda-(1-H(\tau))\lambda^2} = 2^{-\Omega(\lambda^2)}$. Now fix $\mathsf{sk} \in \{0,1\}^{k_{\mathsf{cmb}}}$, u and ρ_{Qry}, and let $\mathsf{Qry}_{\mathsf{sk}}(u;\rho_{\mathsf{Qry}}) = (j',\kappa,cw_0)$. Over the randomness of the lazy-sampled oracle,

$$\Pr\big[j' = j \ \wedge\ \mathsf{Dec}_j(\kappa,cw_0) \neq \bot\big] \ \le\ \Pr[\mathsf{Dec}_j(\kappa,cw_0) \neq \bot] \ \le\ 2^{r_j} \cdot \frac{\mathsf{Vol}_{n_j}(\tau n_j)}{2^{n_j}} \ =\ \mu_\lambda,$$

where the first inequality is trivial, and the second follows from Equation 1. This bound is uniform in (u,ρ_{Qry}). Therefore, $\mathbb{E}_{\mathsf{PRC}_j}[a^{\mathrm{rand}}_{\mathsf{sk},j}] \le \mu_\lambda$.

Define the random variable X_j of above expectation over the choice of the secret key sk, i.e., $X_j := \mathbb{E}_{\mathsf{sk}\leftarrow\{0,1\}^{k_{\mathsf{cmb}}}}[a_{\mathsf{sk},j}]$. Averaging the previous inequality over sk, we obtain $\mathbb{E}_{\mathsf{PRC}_j}[X_j] \le \mu_\lambda$. Fix any function $\eta(\lambda) \in (0,1]$. By Markov's inequality,

$$\Pr_{\mathsf{PRC}_j}\left[X_j > \frac{\mu_\lambda}{\eta(\lambda)}\right] \ \le\ \eta(\lambda).$$

Hence, with probability at least $1-\eta(\lambda)$ over the lazy sampling of PRC_j, we have $X_j \le \frac{\mu_\lambda}{\eta(\lambda)}$. Condition on such a fixing of PRC_j. Then for every function $t(\lambda) > 0$, another application of Markov's inequality gives

$$\Pr_{\mathsf{sk}\leftarrow\{0,1\}^{k_{\mathsf{cmb}}}}\big[a^{\mathrm{rand}}_{\mathsf{sk},j} > t(\lambda)\big] \ \le\ \frac{X_j}{t(\lambda)} \ \le\ \frac{\mu_\lambda}{\eta(\lambda)\,t(\lambda)}.$$

This proves the claim. □

Remark 5.6. *For example, if we set $\eta(\lambda) = 2^{-\lambda}$ and $t(\lambda) = 1/\mathrm{poly}(\lambda)$, then the lemma gives*

$$\Pr_{\mathsf{sk}\leftarrow\{0,1\}^{k_{\mathsf{cmb}}}}\big[a^{\mathrm{rand}}_{\mathsf{sk},j} > 1/\mathrm{poly}(\lambda)\big] \ \le\ \frac{2^{-\Omega(\lambda^2)}}{2^{-\lambda}/\mathrm{poly}(\lambda)} \ =\ 2^{-\Omega(\lambda)},$$

which is negligible in λ. Thus, except with exponentially small probability over the sampled oracle, the branch-j random-world acceptance mass is at most inverse-polynomial for all but a negligible fraction of keys.

5.3 Combining the Distinguishing Attacks

In this section, we will now combine the last two sections. If an inverse-polynomial gap between the real and the random world leads to a distinguisher, and if in the random world, the decoding of the first query is $\bot$ with high probability, then also in the real world, i.e., for a valid codeword of the combined scheme, the decoding of the first query is $\bot$ with high probability.

Let $\varepsilon(\lambda)$ be any inverse-polynomial function. Independently sample $\mathsf{PRC}_1,\ldots,\mathsf{PRC}_n$ according to the lazy-sampled distributions from Section 5.2.

Lemma 5.7. *With probability at least $1-2^{-\lambda+n}$ over the joint lazy sampling of $(\mathsf{PRC}_1,\ldots,\mathsf{PRC}_n)$, the following holds: Suppose that for each $j \in [n]$, for a $1-\varepsilon(\lambda)$ fraction of keys $\mathsf{sk} \in \{0,1\}^{k_{\mathsf{cmb}}}$,*

$$\big|p^{\mathrm{real}}_{\mathsf{sk},j} - p^{\mathrm{rand}}_{\mathsf{sk},j}\big| \le \frac{\varepsilon(\lambda)}{2n}.$$

Then for a $1-\varepsilon(\lambda)$ fraction of keys $\mathsf{sk} \in \{0,1\}^{k_{\mathsf{cmb}}}$,

$$\Pr_{\substack{y\leftarrow\mathsf{Enc}_{\mathsf{cmb}}(\mathsf{sk})\\ \rho_{\mathsf{Qry}}}}\big[\mathsf{Dec}_{j'}(\kappa,cw_0) = \bot \mid (j',\kappa,cw_0) = \mathsf{Qry}_{\mathsf{sk}}(y;\rho_{\mathsf{Qry}})\big] \ \ge\ 1-\varepsilon(\lambda).$$

As any polynomial gap leads to a distinguisher, this lemma, together with Lemma 5.2, directly implies Theorem 5.1. We will now first prove the theorem, and then give the proof for Lemma 5.7.

of Theorem 5.1. Assume that

$$\Pr_{\substack{y\leftarrow\mathsf{Enc}_{\mathsf{cmb}}(\mathsf{sk})\\ \rho_{\mathsf{Qry}}}}\left[\mathsf{Dec}_{j'}(\kappa,cw_0)=\perp \mid (j',\kappa,cw_0)=\mathsf{Qry}_{\mathsf{sk}}(y;\rho_{\mathsf{Qry}})\right] \;<\; 1-\varepsilon(\lambda).$$

Then, by Lemma 5.7, we have a polynomial acceptance gap, i.e., $\left|p^{\mathrm{real}}_{\mathsf{sk},j}-p^{\mathrm{rand}}_{\mathsf{sk},j}\right|>\frac{\varepsilon(\lambda)}{2n}$. This, by Lemma 5.2, implies a distinguisher against the pseudorandomness of the combiner, contradicting the assumption. □

of Lemma 5.7. Since exactly one branch $j'\in[n]$ is chosen by $\mathsf{Qry}_{\mathsf{sk}}$ on every input, we have

$$\sum_{j\in[n]} p^{\mathrm{real}}_{\mathsf{sk},j} \;=\; \Pr_{\substack{y\leftarrow\mathsf{Enc}_{\mathsf{cmb}}(\mathsf{sk})\\ \rho_{\mathsf{Qry}}}}\left[\mathsf{Dec}_{j'}(\kappa,cw_0)=\perp \mid (j',\kappa,cw_0)=\mathsf{Qry}_{\mathsf{sk}}(y;\rho_{\mathsf{Qry}})\right]$$

and $\sum_{j\in[n]} a^{\mathrm{rand}}_{\mathsf{sk},j} \;=\; 1-\sum_{j\in[n]} p^{\mathrm{rand}}_{\mathsf{sk},j}$. Set $\eta(\lambda):=2^{-\lambda}$. By Lemma 5.5, for each $j\in[n]$, with probability at least $1-\eta(\lambda)$ over the lazy sampling of PRC_j, we have

$$\Pr_{\mathsf{sk}\leftarrow\{0,1\}^{k_{\mathsf{cmb}}}}\left[a^{\mathrm{rand}}_{\mathsf{sk},j}>\varepsilon(\lambda)/2n\right] \;\le\; \frac{2n\mu_\lambda}{\eta(\lambda)\,\varepsilon(\lambda)}.$$

Since $\mu_\lambda=2^{-\Omega(\lambda^2)}$, $\eta(\lambda)=2^{-\lambda}$, and $\varepsilon(\lambda)$ is inverse-polynomial, the right-hand side is negligible in λ. By independence of the lazy sampling for $j\in[n]$ and a union bound, with probability at least $1-n\eta(\lambda)=1-\mathrm{negl}(\lambda)$ over the joint lazy sampling of $(\mathsf{PRC}_1,\ldots,\mathsf{PRC}_n)$, the following bound holds for all j simultaneously:

$$\Pr_{\mathsf{sk}\leftarrow\{0,1\}^{k_{\mathsf{cmb}}}}\left[a^{\mathrm{rand}}_{\mathsf{sk},j}>\varepsilon(\lambda)/2n\right]\le\mathrm{negl}(\lambda)$$

Condition on this event from now on. For each $j\in[n]$, let $S_1^{(j)}$ be the set of keys sk such that $\left|p^{\mathrm{real}}_{\mathsf{sk},j}-p^{\mathrm{rand}}_{\mathsf{sk},j}\right|\le\frac{\varepsilon(\lambda)}{2n}$, and let $S_2^{(j)}$ be the set of keys sk such that $a^{\mathrm{rand}}_{\mathsf{sk},j}\le\frac{\varepsilon(\lambda)}{4}$. By hypothesis, $|S_1^{(j)}|/2^{k_{\mathsf{cmb}}}\ge 1-\mathrm{negl}(\lambda)$, and by the previous paragraph, $|S_2^{(j)}|/2^{k_{\mathsf{cmb}}}\ge 1-\mathrm{negl}(\lambda)$. Therefore, by a union bound over $j\in[n]$,

$$\Pr_{\mathsf{sk}\leftarrow\{0,1\}^{k_{\mathsf{cmb}}}}\left[\mathsf{sk}\in\bigcap_j S_1^{(j)}\cap\bigcap_j S_2^{(j)}\right] \;\ge\; 1-\mathrm{negl}(\lambda).$$

Fix any sk in this intersection. Then $\sum_j p^{\mathrm{rand}}_{\mathsf{sk},j} \;=\; 1-\sum_j a^{\mathrm{rand}}_{\mathsf{sk},j} \;\ge\; 1-\frac{\varepsilon(\lambda)}{2}$. Moreover, for all j, $p^{\mathrm{real}}_{\mathsf{sk},j} \;\ge\; p^{\mathrm{rand}}_{\mathsf{sk},j}-\frac{\varepsilon(\lambda)}{2n}$. Summing these inequalities gives

$$\sum_j p^{\mathrm{real}}_{\mathsf{sk},j} \;\ge\; \sum_j p^{\mathrm{rand}}_{\mathsf{sk},j}-\frac{\varepsilon(\lambda)}{2} \;\ge\; 1-\varepsilon(\lambda).$$

Recalling the identity

$$\sum_j p^{\mathrm{real}}_{\mathsf{sk},j} \;=\; \Pr_{\substack{y\leftarrow\mathsf{Enc}_{\mathsf{cmb}}(\mathsf{sk})\\ \rho_{\mathsf{Qry}}}}\left[\mathsf{Dec}_{j'}(\kappa,cw_0)=\perp \mid (j',\kappa,cw_0)=\mathsf{Qry}_{\mathsf{sk}}(y;\rho_{\mathsf{Qry}})\right],$$

we obtain the desired conclusion. Since this holds for every sk in a set of density $1-\mathrm{negl}(\lambda)$, the theorem follows. □

6 Compiling Out Encode Queries

Assume throughout this section that the first oracle query made by $\mathsf{Dec}_{\mathsf{cmb}}$ is always an *encode* query. Unlike a first decode query, such a query can genuinely matter on an honest codeword, because it may hit a branch encoding entry already used by $\mathsf{Enc}_{\mathsf{cmb}}$. We therefore use the heavy-query framework of [DMSR26, GGW25]: all globally heavy first encode queries are learned in advance and stored in an augmented key, while globally light queries can be resampled locally except with small probability. Our proof is a direct specialization of this framework to the lazy-sampled branch PRCs from Definition 4.2. The only external ingredients we need are the exceptionality bound for uniform inputs and its transfer to pseudorandom codewords, which we restate below in our notation.

Fix constant $\tau \in (0, 1/4)$. Let $\mathsf{PRC}_{\mathsf{cmb}} = (\mathsf{Enc}_{\mathsf{cmb}}, \mathsf{Dec}_{\mathsf{cmb}})$ be a zero-bit secret-key pseudorandom code with secret key length k_{cmb} and codeword length n_{cmb}. Independently sample $\mathsf{PRC}_1 = (\mathsf{Enc}_1, \mathsf{Dec}_1), \ldots, \mathsf{PRC}_n = (\mathsf{Enc}_n, \mathsf{Dec}_n)$ according to Definition 4.2, using branch decoding radius τ. Suppose that for every $\mathsf{sk} \in \{0,1\}^{k_{\mathsf{cmb}}}$, every input $x \in \{0,1\}^{n_{\mathsf{cmb}}}$, and every choice of internal coins ρ_{Qry}, the first oracle query made by $\mathsf{Dec}_{\mathsf{cmb}}(\mathsf{sk}, x)$ is an encode query of the form

$$\mathsf{Qry}_{\mathsf{sk}}(x; \rho_{\mathsf{Qry}}) = (j', \kappa, r),$$

where

$$(j', \kappa, r) \in \bigcup_{j \in [n]} \big(j \times \{0,1\}^{k_j} \times \{0,1\}^{r_j}\big),$$

after which the decoder receives

$$cw_0 := \mathsf{Enc}_{j'}(\kappa; r).$$

Note that $\mathsf{Qry}_{\mathsf{sk}}$ is efficiently computable from the description of $\mathsf{Dec}_{\mathsf{cmb}}$.

Theorem 6.1 (Compiling out a single encode query)**.** *Let $M_{\mathsf{cmb}}(\lambda)$ be an upper bound on the total number of encode queries made by $\mathsf{Enc}_{\mathsf{cmb}}$ in any execution, and let $l(\lambda)$ be an upper bound on the total number of oracle queries made by $\mathsf{Dec}_{\mathsf{cmb}}$. Then for every inverse-polynomial function $\kappa^\star(\lambda) > 0$, there exist:*

- *an unbounded learner $\mathsf{LearnEnc}_{\mathsf{cmb}}$, and*
- *an efficient randomized decoder $\mathsf{ADec}_{\mathsf{cmb}}$ that makes at most $l(\lambda) - 1$ oracle queries,*

such that the following holds. Let

$$\delta(\lambda) := \frac{\kappa^\star(\lambda)}{2M_{\mathsf{cmb}}(\lambda)} \qquad \text{and} \qquad \varepsilon(\lambda) := \left(\frac{\kappa^\star(\lambda)\delta(\lambda)^2}{6}\right)^{\frac{1+\tau^2}{1-\tau^2}}.$$

Then

$$\Pr\left[\mathsf{ADec}_{\mathsf{cmb}}(\mathsf{sk}, \mathsf{aux}, c+e) \neq \mathsf{Dec}_{\mathsf{cmb}}(\mathsf{sk}, c+e) \;:\; \begin{array}{r} (\mathsf{PRC}_1, \ldots, \mathsf{PRC}_n) \\ \mathsf{sk} \leftarrow \{0,1\}^{k_{\mathsf{cmb}}} \\ c \leftarrow \mathsf{Enc}_{\mathsf{cmb}}(\mathsf{sk}) \\ e \leftarrow \mathsf{Ber}_\tau^{n_{\mathsf{cmb}}} \\ \mathsf{aux} \leftarrow \mathsf{LearnEnc}_{\mathsf{cmb}}(\mathsf{sk}) \end{array}\right] \leq \kappa^\star(\lambda).$$

Moreover, $\mathsf{LearnEnc}_{\mathsf{cmb}}$ stores at most $1/\varepsilon(\lambda)$ query-response pairs.

Auxiliary tools from [DMSR26]. The following theorem restates Theorem 8 of [DMSR26] in our notation.

Theorem 6.2 (Exceptionality bound)**.** *Fix integers $n, r > 0$, a noise parameter $\rho \in (0,1)$, a finite set Q, and a query function $\mathsf{Qry} : \{0,1\}^n \times \{0,1\}^r \to Q$. Define the global weight function $w : Q \to [0,1]$ by*

$$w(q) \;:=\; \Pr_{\substack{u \leftarrow \{0,1\}^n \\ \rho_{\mathsf{Qry}} \leftarrow \{0,1\}^r}} \big[\mathsf{Qry}(u; \rho_{\mathsf{Qry}}) = q\big],$$

and for every $x \in \{0,1\}^n$, define the local weight function $\ell_x : Q \to [0,1]$ by

$$\ell_x(q) \;:=\; \Pr_{\substack{e \leftarrow \mathsf{Ber}_\rho^n \\ \rho_{\mathsf{Qry}} \leftarrow \{0,1\}^r}} \big[\mathsf{Qry}(x+e; \rho_{\mathsf{Qry}}) = q\big].$$

For $0 < \varepsilon < \delta < 1$, let $\mathsf{EXC}_{\mathsf{Qry},\varepsilon,\delta}(x)$ denote the event that there exists $q \in Q$ such that

$$w(q) < \varepsilon \qquad \text{and} \qquad \ell_x(q) \geq \delta.$$

Then for every $t > 0$,

$$\Pr_{u \leftarrow \{0,1\}^n} \big[\mathsf{EXC}_{\mathsf{Qry},\varepsilon,\delta}(u)\big] \;\leq\; \frac{\varepsilon^{\frac{1+t}{1+\rho^2 t}-1}}{\delta^{1+t}}.$$

The next lemma restates the empty-auxiliary case of Lemma 4 of [DMSR26].

Lemma 6.3 (Transfer to pseudorandom codewords)**.** *Let $\mathsf{PRC} = (\mathsf{Enc}, \mathsf{Dec})$ be a zero-bit secret-key pseudorandom code with secret key length k and codeword length n. Fix a noise parameter $\rho \in (0,1)$, a finite set Q, and for each key $\mathsf{sk} \in \{0,1\}^k$, a query function $\mathsf{Qry}_{\mathsf{sk}} : \{0,1\}^n \times \{0,1\}^r \to Q$. For $0 < \varepsilon < \delta < 1$, define w_{sk}, $\ell_{\mathsf{sk},x}$, and $\mathsf{EXC}_{\mathsf{sk},\varepsilon,\delta}(x)$ exactly as in Theorem 6.2, with Qry replaced by $\mathsf{Qry}_{\mathsf{sk}}$. Let*

$$p^\star(\varepsilon, \delta) \;:=\; \frac{\varepsilon^{\frac{1-\rho^2}{1+\rho^2}}}{\delta^2}.$$

If PRC is pseudorandom, then

$$\Pr\left[\mathsf{EXC}_{\mathsf{sk},\varepsilon,\delta}(c) \;:\; \begin{array}{c} \mathsf{sk} \leftarrow \{0,1\}^k \\ c \leftarrow \mathsf{Enc}(\mathsf{sk}) \end{array}\right] \;\leq\; 3p^\star(\varepsilon, \delta).$$

Proof. **Setup.** Fix an inverse-polynomial function $\kappa^\star(\lambda) > 0$, and let

$$Q^{\mathsf{Enc}} \;:=\; \bigcup_{j \in [n]} \big(j \times \{0,1\}^{k_j} \times \{0,1\}^{r_j}\big)$$

denote the space of possible first encode queries made by $\mathsf{Dec}_{\mathsf{cmb}}$. For every key $\mathsf{sk} \in \{0,1\}^{k_{\mathsf{cmb}}}$, define the global weight function $w_{\mathsf{sk}} : Q^{\mathsf{Enc}} \to [0,1]$ by

$$w_{\mathsf{sk}}(q) \;:=\; \Pr_{\substack{u \leftarrow \{0,1\}^{n_{\mathsf{cmb}}} \\ \rho_{\mathsf{Qry}}}} \big[\mathsf{Qry}_{\mathsf{sk}}(u; \rho_{\mathsf{Qry}}) = q\big],$$

and for every $x \in \{0,1\}^{n_{\mathsf{cmb}}}$, define the local weight function $\ell_{\mathsf{sk},x} : Q^{\mathsf{Enc}} \to [0,1]$ by

$$\ell_{\mathsf{sk},x}(q) \;:=\; \Pr_{\substack{e \leftarrow \mathsf{Ber}_\tau^{n_{\mathsf{cmb}}} \\ \rho_{\mathsf{Qry}}}} \big[\mathsf{Qry}_{\mathsf{sk}}(x+e;\rho_{\mathsf{Qry}}) = q\big].$$

For parameters $0 < \varepsilon < \delta < 1$, let $\mathsf{EXC}_{\mathsf{sk},\varepsilon,\delta}(x)$ denote the event that x is (ε,δ)-exceptional with respect to $\mathsf{Qry}_{\mathsf{sk}}$, namely, that there exists $q \in Q^{\mathsf{Enc}}$ such that

$$w_{\mathsf{sk}}(q) < \varepsilon \qquad \text{and} \qquad \ell_{\mathsf{sk},x}(q) \geq \delta.$$

Further define

$$p^\star(\lambda) \;:=\; \frac{\varepsilon(\lambda)^{\frac{1-\tau^2}{1+\tau^2}}}{\delta(\lambda)^2}.$$

By construction,

$$3p^\star(\lambda) = 3 \cdot \frac{\varepsilon(\lambda)^{\frac{1-\tau^2}{1+\tau^2}}}{\delta(\lambda)^2} = \frac{\kappa^\star(\lambda)}{2}.$$

Construction of the learner and augmented decoder. For a fixed sampled tuple $(\mathsf{PRC}_1,\ldots,\mathsf{PRC}_n)$ and a fixed key sk, define the set of globally $\varepsilon(\lambda)$-heavy queries

$$\widetilde{Q}_{\mathsf{sk}} \;:=\; \big\{q \in Q^{\mathsf{Enc}} : w_{\mathsf{sk}}(q) \geq \varepsilon(\lambda)\big\}.$$

Since $\sum_{q \in Q^{\mathsf{Enc}}} w_{\mathsf{sk}}(q) = 1$, we have $|\widetilde{Q}_{\mathsf{sk}}| \leq 1/\varepsilon(\lambda)$. The learner $\mathsf{LearnEnc}_{\mathsf{cmb}}$ is defined as follows on input sk:

1. Compute $w_{\mathsf{sk}}(q)$ exactly for every $q \in Q^{\mathsf{Enc}}$.
2. Form the set $\widetilde{Q}_{\mathsf{sk}}$ of all globally $\varepsilon(\lambda)$-heavy queries.
3. For every $q = (j',\kappa,r) \in \widetilde{Q}_{\mathsf{sk}}$, query the sampled branch encoder and store the pair $\big(q, \mathsf{Enc}_{j'}(\kappa;r)\big)$.
4. Output the resulting list as aux.

The augmented decoder $\mathsf{ADec}_{\mathsf{cmb}}$ is defined as follows on input $(\mathsf{sk},\mathsf{aux},y)$, where $y \in \{0,1\}^{n_{\mathsf{cmb}}}$:

1. Sample the same internal randomness that $\mathsf{Dec}_{\mathsf{cmb}}$ would use, in particular the same coins ρ_{Qry} for the first-query procedure.
2. Compute $q = \mathsf{Qry}_{\mathsf{sk}}(y;\rho_{\mathsf{Qry}}) = (j,\kappa,r)$.
3. If q appears in the stored list aux, return the stored answer cw_0.
4. Otherwise, sample $cw \leftarrow \{0,1\}^{n_j}$, where n_j is the codeword length of branch j, and use this as the answer to the first encode query.
5. For any subsequent query $(\mathsf{Op}',j',\kappa',\mathsf{inp}')$ of $\mathsf{Dec}_{\mathsf{cmb}}$, respond as follows:
   (a) if $(\mathsf{Op}',j',\kappa',\mathsf{inp}') = (\mathsf{Enc},j,\kappa,r)$, then return cw to $\mathsf{Dec}_{\mathsf{cmb}}$.
   (b) if $\mathsf{Op}' = \mathsf{Dec}$ and $j' = j$ and $\mathsf{Ham}(\mathsf{inp}',cw) \leq \tau n_j$, then return $\top$.

(c) otherwise, query the appropriate oracle and return its response.

By construction, $\mathsf{ADec}_{\mathsf{cmb}}$ makes at most $l(\lambda) - 1$ oracle queries.

Agreement outside a bad event. For a fixed key sk and a fixed honest codeword $c \leftarrow \mathsf{Enc}_{\mathsf{cmb}}(\mathsf{sk})$, let $Q_{\mathsf{Enc}}(\mathsf{sk}, c) \subseteq Q^{\mathsf{Enc}}$ denote the set of all encode queries made by $\mathsf{Enc}_{\mathsf{cmb}}$ during the computation of c. Since $\mathsf{Enc}_{\mathsf{cmb}}$ is efficient,

$$|Q_{\mathsf{Enc}}(\mathsf{sk}, c)| \leq M_{\mathsf{cmb}}(\lambda) = \mathrm{poly}(\lambda).$$

Now couple the executions of $\mathsf{Dec}_{\mathsf{cmb}}(\mathsf{sk}, c + e)$ and $\mathsf{ADec}_{\mathsf{cmb}}(\mathsf{sk}, \mathsf{aux}, c + e)$ so that they use the same internal randomness and the same coins ρ_{Qry} for the first-query procedure $\mathsf{Qry}_{\mathsf{sk}}$. Let $q = \mathsf{Qry}_{\mathsf{sk}}(c + e; \rho_{\mathsf{Qry}})$. Define the bad event

$$\mathsf{BAD}_{\mathsf{sk},c,e} \;:=\; \Big\{ q \notin \widetilde{Q}_{\mathsf{sk}} \;\wedge\; q \in Q_{\mathsf{Enc}}(\mathsf{sk}, c) \Big\}.$$

That is, $\mathsf{BAD}_{\mathsf{sk},c,e}$ is the event that the first decoder encode query is globally light but is nevertheless one of the actual encode queries made by the honest combiner encoder while generating c.

We claim that, conditioned on $\neg\mathsf{BAD}_{\mathsf{sk},c,e}$, the output distributions of $\mathsf{Dec}_{\mathsf{cmb}}(\mathsf{sk}, c + e)$ and $\mathsf{ADec}_{\mathsf{cmb}}(\mathsf{sk}, \mathsf{aux}, c + e)$ are identical. Indeed, there are two cases:

1. If $q \in \widetilde{Q}_{\mathsf{sk}}$, then $\mathsf{LearnEnc}_{\mathsf{cmb}}$ has stored the exact value of the corresponding branch encode query, so $\mathsf{ADec}_{\mathsf{cmb}}$ feeds back precisely the same answer as $\mathsf{Dec}_{\mathsf{cmb}}$.
2. If $q \notin \widetilde{Q}_{\mathsf{sk}}$ and $q \notin Q_{\mathsf{Enc}}(\mathsf{sk}, c)$, then the queried table entry $\mathsf{Enc}_j(\kappa; r)$ was not used anywhere during the honest encoding of c. Since the branch PRCs are lazy-sampled as in Definition 4.2, this table entry is uniformly random and independent of the encoder transcript. Therefore, replacing it by a freshly sampled uniform string of the same length, and using that value consistently in any future identical query, yields exactly the same distribution.

Since the two executions are otherwise identical, it follows that

$$\Pr\big[\mathsf{ADec}_{\mathsf{cmb}}(\mathsf{sk}, \mathsf{aux}, c + e) \neq \mathsf{Dec}_{\mathsf{cmb}}(\mathsf{sk}, c + e)\big] \;\leq\; \Pr[\mathsf{BAD}_{\mathsf{sk},c,e}].$$

Averaging over all randomness, it therefore suffices to bound the probability of $\mathsf{BAD}_{\mathsf{sk},c,e}$.

Bounding the bad event via exceptionality. We first invoke Theorem 6.2 with $\rho = \tau$ and $t = 1$, applied to the query family $\mathsf{Qry}_{\mathsf{sk}}$. This yields the uniform-word exceptionality bound

$$\Pr_{u \leftarrow \{0,1\}^{n_{\mathsf{cmb}}}}\big[\mathsf{EXC}_{\mathsf{sk},\varepsilon,\delta}(u)\big] \;\leq\; \frac{\varepsilon^{\frac{1-\tau^2}{1+\tau^2}}}{\delta^2} = p^\star(\lambda).$$

Next, we apply Lemma 6.3 to the combiner PRC $\mathsf{PRC}_{\mathsf{cmb}}$, with $\rho = \tau$ and query function $\mathsf{Qry}_{\mathsf{sk}}$. Since $\mathsf{PRC}_{\mathsf{cmb}}$ is pseudorandom, we obtain

$$\Pr\left[\mathsf{EXC}_{\mathsf{sk},\varepsilon,\delta}(c) \;:\; \begin{array}{c}(\mathsf{PRC}_1, \ldots, \mathsf{PRC}_n) \\ \mathsf{sk} \leftarrow \{0,1\}^{k_{\mathsf{cmb}}} \\ c \leftarrow \mathsf{Enc}_{\mathsf{cmb}}(\mathsf{sk})\end{array}\right] \;\leq\; 3p^\star(\lambda) = \frac{\kappa^\star(\lambda)}{2}.$$

Now condition on the event $\neg\mathsf{EXC}_{\mathsf{sk},\varepsilon,\delta}(c)$. Then for every query $q \in Q^{\mathsf{Enc}}$ with $w_{\mathsf{sk}}(q) < \varepsilon(\lambda)$, we have

$$\ell_{\mathsf{sk},c}(q) < \delta(\lambda),$$

that is,

$$\Pr_{\substack{e \leftarrow \mathsf{Ber}_\tau^{n_{\mathsf{cmb}}} \\ \rho_{\mathsf{Qry}}}} \big[\mathsf{Qry}_{\mathsf{sk}}(c+e;\rho_{\mathsf{Qry}}) = q\big] < \delta(\lambda).$$

Since every query in $Q_{\mathsf{Enc}}(\mathsf{sk},c) \setminus \widetilde{Q}_{\mathsf{sk}}$ is globally light, a union bound gives

$$\begin{aligned}
\Pr[\mathsf{BAD}_{\mathsf{sk},c,e} \mid \neg\mathsf{EXC}_{\mathsf{sk},\varepsilon,\delta}(c)] \;&\le\; \sum_{q \in Q_{\mathsf{Enc}}(\mathsf{sk},c)\setminus\widetilde{Q}_{\mathsf{sk}}} \Pr_{\substack{e \leftarrow \mathsf{Ber}_\tau^{n_{\mathsf{cmb}}} \\ \rho_{\mathsf{Qry}}}} \big[\mathsf{Qry}_{\mathsf{sk}}(c+e;\rho_{\mathsf{Qry}}) = q\big] \\
&\le\; |Q_{\mathsf{Enc}}(\mathsf{sk},c)| \cdot \delta(\lambda) \;\le\; M_{\mathsf{cmb}}(\lambda)\delta(\lambda) \\
&=\; \frac{\kappa^\star(\lambda)}{2}.
\end{aligned}$$

Combining the two bounds,

$$\begin{aligned}
\Pr[\mathsf{BAD}_{\mathsf{sk},c,e}] &\le \Pr\big[\mathsf{EXC}_{\mathsf{sk},\varepsilon,\delta}(c)\big] + \Pr\big[\mathsf{BAD}_{\mathsf{sk},c,e} \mid \neg\mathsf{EXC}_{\mathsf{sk},\varepsilon,\delta}(c)\big] \\
&\le 3p^\star(\lambda) + M_{\mathsf{cmb}}(\lambda)\delta(\lambda) \\
&= \frac{\kappa^\star(\lambda)}{2} + \frac{\kappa^\star(\lambda)}{2} = \kappa^\star(\lambda).
\end{aligned}$$

Therefore,

$$\Pr\left[\begin{array}{l} \mathsf{ADec}_{\mathsf{cmb}}(\mathsf{sk},\mathsf{aux},c+e) \neq \mathsf{Dec}_{\mathsf{cmb}}(\mathsf{sk},c+e) \;:\; \end{array}\begin{array}{r} (\mathsf{PRC}_1,\ldots,\mathsf{PRC}_n) \\ \mathsf{sk} \leftarrow \{0,1\}^{k_{\mathsf{cmb}}} \\ c \leftarrow \mathsf{Enc}_{\mathsf{cmb}}(\mathsf{sk}) \\ e \leftarrow \mathsf{Ber}_\tau^{n_{\mathsf{cmb}}} \\ \mathsf{aux} \leftarrow \mathsf{LearnEnc}_{\mathsf{cmb}}(\mathsf{sk}) \end{array}\right] \;\le\; \kappa^\star(\lambda),$$

as claimed.

Finally, since $|\widetilde{Q}_{\mathsf{sk}}| \le 1/\varepsilon(\lambda)$, the learner stores at most $1/\varepsilon(\lambda)$ query-response pairs. □

7 Compiling Out All Queries

We have now seen how to compile out the first query when it is deterministically an Enc_j-query or a Dec_j-query. In general, however, the query type may depend on the input and the decoder randomness, and compiling out later queries requires a slightly different treatment. We therefore first combine the previous two results into a single lemma that removes the first query independently of its type, and then extend this to the setting where some queries have already been compiled out. Finally, we iterate this argument until the decoder makes no oracle queries at all, at which point the oracle-free impossibility result of [DMSR26] applies.

We continue to work with the same Bernoulli-noise parameter τ and branch decoding radius τ as in the previous two sections.

To show our first result, compiling out the first query independent of its type, we will redefine $\mathsf{Qry}_{\mathsf{cmb}}$ as

$$\mathsf{Qry}_{\mathsf{sk}}(x; \rho_{\mathsf{Qry}}) = (\mathsf{Op}, j', \kappa, \mathsf{inp}),$$

the first query made by $\mathsf{Dec}_{\mathsf{cmb}}(\mathsf{sk}, x)$, where $\mathsf{Op} \in \{\mathsf{Enc}, \mathsf{Dec}\}$ denotes the operation, ρ_{Qry} denotes the internal randomness of Qry, and $\mathsf{Dec}_{\mathsf{cmb}}$ then queries $\mathsf{Op}'_j(\kappa, \mathsf{inp})$ (where inp is the randomness or the codeword, depending on the operation).

Lemma 7.1 (Compiling out first query)**.** *Let $l(\lambda)$ be the maximum number of Encode or Decode queries by $\mathsf{Dec}_{\mathsf{cmb}}$. Then for every inverse-polynomial function $\kappa^\star(\lambda) > 0$, there exist:*

- *an unbounded learner $\mathsf{LearnEnc}_{\mathsf{cmb}}$, and*
- *an efficient randomized decoder $\mathsf{ADec}_{\mathsf{cmb}}$ that makes at most $l(\lambda) - 1$ oracle queries,*

such that the following holds:

$$\Pr\left[\mathsf{ADec}_{\mathsf{cmb}}(\mathsf{sk}, \mathsf{aux}, c+e) \neq \mathsf{Dec}_{\mathsf{cmb}}(\mathsf{sk}, c+e) \ : \begin{array}{r} (\mathsf{PRC}_1, \ldots, \mathsf{PRC}_n) \\ \mathsf{sk} \leftarrow \{0,1\}^{k_{\mathsf{cmb}}} \\ c \leftarrow \mathsf{Enc}_{\mathsf{cmb}}(\mathsf{sk}) \\ e \leftarrow \mathsf{Ber}_\tau^{n_{\mathsf{cmb}}} \\ \mathsf{aux} \leftarrow \mathsf{LearnEnc}_{\mathsf{cmb}}(\mathsf{sk}) \end{array}\right] \leq \kappa^\star(\lambda).$$

Moreover, $\mathsf{LearnEnc}_{\mathsf{cmb}}$ stores polynomially many query-response pairs.

Proof. The learner $\mathsf{LearnEnc}_{\mathsf{cmb}}$ is defined as follows on input sk:

1. Redefine $w_{\mathsf{sk}}(q)$ as

$$w_{\mathsf{sk}}(q) \ := \Pr_{\substack{u \leftarrow \{0,1\}^{n_{\mathsf{cmb}}} \\ \rho_{\mathsf{Qry}}}} \left[\mathsf{Qry}_{\mathsf{sk}}(u; \rho_{\mathsf{Qry}}) = (\mathsf{Enc}, q) \mid \mathsf{Qry}_{\mathsf{sk}}(u; \rho_{\mathsf{Qry}}) = (\mathsf{Enc}, \cdot)\right],$$

   i.e., we condition on the query being an Enc-query.

2. Compute aux as in Theorem 6.1 and return it.

The augmented decoder $\mathsf{ADec}_{\mathsf{cmb}}$ is defined as follows on input $(\mathsf{sk}, \mathsf{aux}, y)$, where $y \in \{0,1\}^{n_{\mathsf{cmb}}}$:

1. Sample the same internal randomness that $\mathsf{Dec}_{\mathsf{cmb}}$ would use, in particular the same coins ρ_{Qry} for the first-query procedure.
2. Compute
$$\mathsf{Qry}_{\mathsf{sk}}(y; \rho_{\mathsf{Qry}}) = (\mathsf{Op}, q = (j, \kappa, \mathsf{inp})).$$
3. If $\mathsf{Op} = \mathsf{Dec}$, set $\mathsf{ans} \leftarrow \perp$.
4. If $\mathsf{Op} = \mathsf{Enc}$ and q appears in the stored list aux, set ans to the stored answer cw_0.
5. If $\mathsf{Op} = \mathsf{Enc}$ and q does not appear in aux, sample $\mathsf{ans} \leftarrow \{0,1\}^{n_j}$, where n_j is the codeword length of branch j, and use this as the answer to the first encode query.

6. For subsequent queries $(\mathsf{Op}', j', \kappa', \mathsf{inp}')$ of $\mathsf{Dec}_{\mathsf{cmb}}$,
   (a) if $(\mathsf{Op}', j', \kappa', \mathsf{inp}') = (\mathsf{Op}, j, \kappa, \mathsf{inp})$, then return ans to $\mathsf{Dec}_{\mathsf{cmb}}$.
   (b) if $\mathsf{Op}' = \mathsf{Dec}$ and $j' = j$ and $\mathsf{Op} = \mathsf{Enc}$ and $\mathsf{Ham}(\mathsf{inp}', \mathsf{ans}) \leq \tau n_j$, then return $\top$.
   (c) otherwise, query the appropriate oracle and return its response.

Condition on the event that the first query type is Dec. Then $\mathsf{ADec}_{\mathsf{cmb}}$ behaves exactly like $\mathsf{Dec}_{\mathsf{cmb}}$, except that it answers the first branch decode query by $\perp$ and thereafter answers any repetition of that same query consistently. By Theorem 5.1, this changes the output with probability at most $\kappa^\star(\lambda)$.

Condition on the event that the first query type is Enc. Then the learner and simulator above are exactly the ones from Theorem 6.1, except that we have repackaged the answer to the first query as ans. The same analysis therefore shows that the output changes with probability at most $\kappa^\star(\lambda)$.

Applying the law of total probability over the two possible first-query types proves the lemma. □

We now know how to remove the first query, regardless of whether it is a Dec_j-query or an Enc_j-query. The next step is to apply this recursively, removing one query at a time. If a previous query was a Dec_j-query, this is easy, since that query is simply replaced by $\perp$. If a previous query was an Enc_j-query, we have to preserve consistency because a later Dec_j-query may be asked on the corresponding branch codeword. Accordingly, we now assume that we already have an augmented decoder $(\mathsf{LearnEnc}_{\mathsf{cmb}}, \mathsf{ADec}_{\mathsf{cmb}})$, and we let $\mathsf{Qry}_{\mathsf{sk}}$ denote the function that returns the first remaining branch-oracle query of $\mathsf{ADec}_{\mathsf{cmb}}$.

Lemma 7.2 (Later Queries)**.** *Let $\mathsf{PRC}_{\mathsf{cmb}} = (\mathsf{Enc}_{\mathsf{cmb}}, \mathsf{Dec}_{\mathsf{cmb}})$ be a $(n, n-1)$-robust PRC combiner in the random PRC model, and let $(\mathsf{LearnEnc}_{\mathsf{cmb}}, \mathsf{ADec}_{\mathsf{cmb}})$ be an augmented decoder for $\mathsf{PRC}_{\mathsf{cmb}}$. Let $l(\lambda)$ be the maximum number of Encode or Decode queries made by $\mathsf{ADec}_{\mathsf{cmb}}$. Then for every inverse-polynomial function $\kappa^\star(\lambda) > 0$, there exist:*

- *an unbounded learner $\mathsf{LearnEnc}'_{\mathsf{cmb}}$, and*
- *an efficient randomized decoder $\mathsf{ADec}'_{\mathsf{cmb}}$ that makes at most $l(\lambda) - 1$ oracle queries,*

such that the following holds:

$$\Pr\left[\mathsf{ADec}'_{\mathsf{cmb}}(\mathsf{sk}, \mathsf{aux}, \mathsf{aux}', c+e) \neq \mathsf{ADec}_{\mathsf{cmb}}(\mathsf{sk}, \mathsf{aux}, c+e) \;:\; \begin{array}{r} (\mathsf{PRC}_1, \ldots, \mathsf{PRC}_n) \\ \mathsf{sk} \leftarrow \{0,1\}^{k_{\mathsf{cmb}}} \\ c \leftarrow \mathsf{Enc}_{\mathsf{cmb}}(\mathsf{sk}) \\ e \leftarrow \mathsf{Ber}_\tau^{n_{\mathsf{cmb}}} \\ (\mathsf{aux}, \mathsf{aux}') \leftarrow \mathsf{LearnEnc}'_{\mathsf{cmb}}(\mathsf{sk}) \end{array}\right] \leq \kappa^\star(\lambda).$$

Moreover, $\mathsf{LearnEnc}'_{\mathsf{cmb}}$ stores polynomially many query-response pairs.

Proof sketch. The learner $\mathsf{LearnEnc}'_{\mathsf{cmb}}$ is defined as follows on input sk:

1. Compute $\mathsf{aux} \leftarrow \mathsf{LearnEnc}_{\mathsf{cmb}}(\mathsf{sk})$.
2. Compute aux' by applying Lemma 7.1 to the decoder $\mathsf{ADec}_{\mathsf{cmb}}$, treating $(\mathsf{sk}, \mathsf{aux})$ as the augmented key.
3. Return $(\mathsf{aux}, \mathsf{aux}')$.

The augmented decoder $\mathsf{ADec}'_{\mathsf{cmb}}$ is defined as follows on input $(\mathsf{sk}, \mathsf{aux}, \mathsf{aux}', y)$, where $y \in \{0,1\}^{n_{\mathsf{cmb}}}$:

1. Sample the same internal randomness that $\mathsf{ADec}_{\mathsf{cmb}}$ would use, in particular the same coins ρ_{Qry} for the first-query procedure.
2. Compute
$$\mathsf{Qry}_{\mathsf{sk}}(y; \rho_{\mathsf{Qry}}) = (\mathsf{Op}, q = (j, \kappa, \mathsf{inp})).$$
This is the first query made by $\mathsf{ADec}_{\mathsf{cmb}}$.
3. If $\mathsf{Op} = \mathsf{Dec}$, set $\mathsf{ans} \leftarrow \bot$.
4. If $\mathsf{Op} = \mathsf{Enc}$ and q appears in the stored list aux', set ans to the stored answer cw_0.
5. If $\mathsf{Op} = \mathsf{Enc}$ and q does not appear in aux', sample $\mathsf{ans} \leftarrow \{0,1\}^{n_j}$, where n_j is the codeword length of branch j, and use this as the answer to the first encode query.
6. For subsequent queries $(\mathsf{Op}', j', \kappa', \mathsf{inp}')$ of $\mathsf{ADec}_{\mathsf{cmb}}$,
   (a) if $(\mathsf{Op}', j', \kappa', \mathsf{inp}') = (\mathsf{Op}, j, \kappa, \mathsf{inp})$, then return ans to $\mathsf{ADec}_{\mathsf{cmb}}$.
   (b) if $\mathsf{Op}' = \mathsf{Dec}$ and $j' = j$ and $\mathsf{Op} = \mathsf{Enc}$ and $\mathsf{Ham}(\mathsf{inp}', \mathsf{ans}) \leq \tau n_j$, then return $\top$.
   (c) otherwise, query the appropriate oracle and return its response.

The point is that Lemma 7.1 can be re-applied to $\mathsf{ADec}_{\mathsf{cmb}}$: once aux is fixed, the decoder still has a well-defined first remaining branch-oracle query. The only extra bookkeeping compared to the first-query case is that previously learned answers in aux must continue to be replayed consistently, and the construction above does exactly that. The same case split as in Lemma 7.1 therefore gives the claimed error bound, and the total stored state remains polynomial because both aux and aux' are polynomially sized.

□

Note that we do not handle separately the case where, for a specific input, $\mathsf{ADec}_{\mathsf{cmb}}$ makes no oracle queries. In that case, $\mathsf{ADec}'_{\mathsf{cmb}}$ simply equals $\mathsf{ADec}_{\mathsf{cmb}}$.

7.1 Oracle-free Decoder

We will now show that for any robust combiner for PRCs, we can build an augmented decoder, which is oracle-free. We will use a hybrid argument for this, where we iteratively use Lemma 7.2 to remove one oracle call at a time.

Lemma 7.3. *Let $\mathsf{PRC}_{\mathsf{cmb}} = (\mathsf{Enc}_{\mathsf{cmb}}, \mathsf{Dec}_{\mathsf{cmb}})$ be a $(n, n-1)$-robust black-box combiner for zero-bit secret-key pseudorandom codes in the random PRC model with secret key length k_{cmb} and codeword length n_{cmb}. Then, for each inverse-polynomial κ^*, there exists an augmented decoder $(\mathsf{LearnEnc}^\star_{\mathsf{cmb}}, \mathsf{ADec}^\star_{\mathsf{cmb}})$, where $\mathsf{ADec}^\star_{\mathsf{cmb}}$ is oracle-free, and*

$$\Pr\left[\begin{array}{l} \mathsf{ADec}^\star_{\mathsf{cmb}}(\mathsf{sk}, \mathsf{aux}, c+e) \neq \\ \mathsf{Dec}_{\mathsf{cmb}}(\mathsf{sk}, c+e) \end{array} : \begin{array}{r} (\mathsf{PRC}_1, \ldots, \mathsf{PRC}_n) \\ \mathsf{sk} \leftarrow \{0,1\}^{k_{\mathsf{cmb}}} \\ c \leftarrow \mathsf{Enc}_{\mathsf{cmb}}(\mathsf{sk}) \\ e \leftarrow \mathsf{Ber}_\tau^{n_{\mathsf{cmb}}} \\ \mathsf{aux} \leftarrow \mathsf{LearnEnc}^\star_{\mathsf{cmb}}(\mathsf{sk}) \end{array}\right] \leq \kappa^*(\lambda).$$

Proof sketch. The proof is a simple hybrid argument, where in each step, we apply Theorem 7.1 or Theorem 7.2. Let ℓ be the maximum number of queries made by $\mathsf{Dec}_{\mathsf{cmb}}$, and let H_0 denote the original decoder. Define H_1 by applying Theorem 7.1 to H_0 with per-step error parameter $\kappa^*(\lambda)/\ell$. For $2 \leq i \leq \ell$, let H_i be the augmented decoder obtained by applying Theorem 7.2 to H_{i-1}, again with per-step error parameter $\kappa^*(\lambda)/\ell$. Then,

$$\begin{aligned}\Pr[H_0 \neq H_\ell] &\leq \sum_{1 \leq i \leq \ell} \Pr[H_{i-1} \neq H_i] \\ &\leq \sum_{1 \leq i \leq \ell} \kappa^*(\lambda)/\ell \\ &= \kappa^*(\lambda).\end{aligned}$$

□

We now show that once the decoder has been made oracle-free, the combined scheme can no longer be pseudorandom. This is a direct adaptation of Theorem 10 of [DMSR26].

Lemma 7.4. *Let* $\mathsf{PRC}_{\mathsf{cmb}} = (\mathsf{Enc}_{\mathsf{cmb}}, \mathsf{Dec}_{\mathsf{cmb}})$ *be a robust combiner for PRCs in the random PRC model, and let* $\mathsf{LearnEnc}, \mathsf{ADec}$ *be an augmented decoder such that the correctness error is at most* $\frac{1}{8}$*. Then, there exists an unbounded distinguisher against the pseudorandomness of* $\mathsf{PRC}_{\mathsf{cmb}}$ *with advantage* $\frac{1}{16} - 2^{-\lambda}$*.*

Proof sketch. This is exactly the point where we invoke the oracle-free impossibility machinery of [DMSR26]. Once ADec makes no oracle queries, the pair $(\mathsf{LearnEnc}, \mathsf{ADec})$ becomes an explicit augmented decoder whose behavior is determined entirely by the augmented key $(\mathsf{sk}, \mathsf{aux})$. The learner stores only polynomially many bits, so the augmented-key space remains finite and can be enumerated by an unbounded distinguisher.

The distinguisher from Theorem 10 of [DMSR26] now applies verbatim: it draws polynomially many challenge samples, adds independent Bernoulli noise to them, and exhaustively searches for an augmented key $(\mathsf{sk}, \mathsf{aux})$ under which ADec accepts noticeably many of the resulting noisy words. In the real world, the true augmented key passes this test with high probability because the correctness error is at most $1/8$. In the random world, the inputs remain uniform even after adding independent noise, and Theorem 10 of [DMSR26] shows that no oracle-free augmented decoder can make these two cases look the same. The resulting distinguishing advantage is $\frac{1}{16} - 2^{-\lambda}$. □

We are now ready to prove the main theorem of this part, showing that no black-box combiner of PRCs can exist:

of Theorem 4.4. Assume such a combiner $\mathsf{PRC}_{\mathsf{cmb}} = (\mathsf{Enc}_{\mathsf{cmb}}, \mathsf{Dec}_{\mathsf{cmb}})$ with correctness error at most $\frac{1}{16}$, which we instantiate with PRC_j. By Lemma 7.3, we can augment the scheme with $(\mathsf{LearnEnc}^\star_{\mathsf{cmb}}, \mathsf{ADec}^\star_{\mathsf{cmb}})$, such that $\mathsf{ADec}^\star$ does not make any queries to the underlying PRCs PRC_j, and by choosing $\kappa^* = \frac{1}{16}$, the augmented scheme has a correctness error of at most $\frac{1}{8}$. Then, however, by Lemma 7.4, we have a distinguisher against the pseudorandomness of the scheme, which is a contradiction to the assumption. □

Acknowledgments

We thank Roman Langrehr for pointing out that our impossibility results rule out black-box constructions of public-key encryption with pseudorandom ciphertexts from IND-CPA public-key encryption. We also thank the anonymous reviewers for valuable comments.

The authors used OpenAI tools to assist with basic language editing of portions of this manuscript. The scientific content, claims, proofs, and conclusions are the authors' own. The resulting work, in its totality, is an accurate representation of the authors' underlying work, and the authors take full responsibility for the veracity and correctness of all material.

References

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

[Aar22] Scott Aaronson. My AI safety lecture for UT effective altruism, 2022. *URL https://scottaaronson.blog*, 2022.

[ABJ+19] Prabhanjan Ananth, Saikrishna Badrinarayanan, Aayush Jain, Nathan Manohar, and Amit Sahai. From fe combiners to secure mpc and back. In *Theory of Cryptography Conference*, pages 199–228. Springer, 2019.

[AGGM06] Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. On basing one-way functions on NP-hardness. In Jon M. Kleinberg, editor, *38th Annual ACM Symposium on Theory of Computing*, pages 701–710, Seattle, WA, USA, May 21–23, 2006. ACM Press.

[AGGM10] Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. Erratum for: on basing one-way functions on NP-hardness. In Leonard J. Schulman, editor, *42nd Annual ACM Symposium on Theory of Computing*, pages 795–796, Cambridge, MA, USA, June 5–8, 2010. ACM Press.

[AJN+16] Prabhanjan Ananth, Aayush Jain, Moni Naor, Amit Sahai, and Eylon Yogev. Universal constructions and robust combiners for indistinguishability obfuscation and witness encryption. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 491–520, Santa Barbara, CA, USA, August 14–18, 2016. Springer Berlin Heidelberg, Germany.

[AJS17] Prabhanjan Ananth, Aayush Jain, and Amit Sahai. Robust transforming combiners from indistinguishability obfuscation to functional encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 91–121. Springer, 2017.

[AS15] Gilad Asharov and Gil Segev. Limits on the power of indistinguishability obfuscation and functional encryption. In Venkatesan Guruswami, editor, *56th Annual Symposium*

on Foundations of Computer Science, pages 191–209, Berkeley, CA, USA, October 17–20, 2015. IEEE Computer Society Press.

[BB04] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 443–459, Santa Barbara, CA, USA, August 15–19, 2004. Springer Berlin Heidelberg, Germany.

[BB15] Andrej Bogdanov and Christina Brzuska. On basing size-verifiable one-way functions on NP-hardness. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 1–6, Warsaw, Poland, March 23–25, 2015. Springer Berlin Heidelberg, Germany.

[BBF13] Paul Baecher, Christina Brzuska, and Marc Fischlin. Notions of black-box reductions, revisited. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 296–315, Bengalore, India, December 1–5, 2013. Springer Berlin Heidelberg, Germany.

[BFKL93] Avrim Blum, Merrick Furst, Michael Kearns, and Richard J Lipton. Cryptographic primitives based on hard learning problems. In *Annual international cryptology conference*, pages 278–291. Springer, 1993.

[BKR23] Andrej Bogdanov, Pravesh K Kothari, and Alon Rosen. Public-key encryption, local pseudorandom generators, and the low-degree method. In *Theory of Cryptography Conference*, pages 268–285. Springer, 2023.

[BKSY11] Zvika Brakerski, Jonathan Katz, Gil Segev, and Arkady Yerukhimovich. Limits on the power of zero-knowledge proofs in cryptographic constructions. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 559–578, Providence, RI, USA, March 28–30, 2011. Springer Berlin Heidelberg, Germany.

[BMM24] Amos Beimel, Tal Malkin, and Noam Mazor. Structural lower bounds on black-box constructions of pseudorandom functions. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part V*, volume 14924 of *Lecture Notes in Computer Science*, pages 459–488, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[BPR+08] Dan Boneh, Periklis A. Papakonstantinou, Charles Rackoff, Yevgeniy Vahlis, and Brent Waters. On the impossibility of basing identity based encryption on trapdoor permutations. In *49th Annual Symposium on Foundations of Computer Science*, pages 283–292, Philadelphia, PA, USA, October 25–28, 2008. IEEE Computer Society Press.

[BT03] Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for NP problems. In *44th Annual Symposium on Foundations of Computer Science*, pages 308–317, Cambridge, MA, USA, October 11–14, 2003. IEEE Computer Society Press.

[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.

[CGG+25] Miranda Christ, Noah Golowich, Sam Gunn, Ankur Moitra, and Daniel Wichs. Improved pseudorandom codes from permuted puzzles. *arXiv preprint arXiv:2512.08918*, 2025.

[DK05] Yevgeniy Dodis and Jonathan Katz. Chosen-ciphertext security of multiple encryption. In *Theory of Cryptography Conference*, pages 188–209. Springer, 2005.

[DM24] Nico Döttling and Tamer Mour. On the black-box complexity of correlation intractability. In *15th Innovations in Theoretical Computer Science Conference (ITCS 2024)*, pages 40–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2024.

[DMSR26] Nico Döttling, Anne Müller, and Mahesh Sreekumar Rajasree. Separating pseudorandom codes from local oracles. In Benny Applebaum and Huijia (Rachel) Lin, editors, *Theory of Cryptography*, pages 225–257, Cham, 2026. Springer Nature Switzerland.

[FFHH26] Sebastian Faller, Marc Fischlin, Julius Hardt, and Julia Hesse. Combining oblivious pseudorandom functions. In Joan Daemen and Emmanuel Thomé, editors, *Advances in Cryptology – EUROCRYPT 2026*, pages 97–127, Cham, 2026. Springer Nature Switzerland.

[FGJ+25] Jaiden Fairoze, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, and Mingyuan Wang. Publicly-detectable watermarking for language models. *IACR Communications in Cryptology*, 1(4), 2025.

[FHNS16] Marc Fischlin, Amir Herzberg, Hod Bin Noon, and Haya Shulman. Obfuscation combiners. Cryptology ePrint Archive, Report 2016/289, 2016.

[FR23] Marc Fischlin and Felix Rohrbach. Searching for ELFs in the cryptographic forest. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023: 21st Theory of Cryptography Conference, Part III*, volume 14371 of *Lecture Notes in Computer Science*, pages 207–236, Taipei, Taiwan, November 29 – December 2, 2023. Springer, Cham, Switzerland.

[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)*, volume 353 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 54:1–54:22, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 276–288, Santa Barbara, CA, USA, August 19–23, 1984. Springer Berlin Heidelberg, Germany.

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

[GHP18] Federico Giacon, Felix Heuer, and Bertram Poettering. Kem combiners. In *IACR International Workshop on Public Key Cryptography*, pages 190–218. Springer, 2018.

[GKM+00] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *41st Annual Symposium on Foundations of Computer Science*, pages 325–335, Redondo Beach, CA, USA, November 12–14, 2000. IEEE Computer Society Press.

[GM24a] Noah Golowich and Ankur Moitra. Edit distance robust watermarks via indexing pseudorandom codes. *Advances in Neural Information Processing Systems*, 37:20645–20693, 2024.

[GM24b] 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.

[GT00] Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *41st Annual Symposium on Foundations of Computer Science*, pages 305–313, Redondo Beach, CA, USA, November 12–14, 2000. IEEE Computer Society Press.

[HHRS07] Iftach Haitner, Jonathan J. Hoch, Omer Reingold, and Gil Segev. Finding collisions in interactive protocols - a tight lower bound on the round complexity of statistically-hiding commitments. In *48th Annual Symposium on Foundations of Computer Science*, pages 669–679, Providence, RI, USA, October 20–23, 2007. IEEE Computer Society Press.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[HKN+05a] Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On robust combiners for oblivious transfer and other primitives. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 96–113. Springer, 2005.

[HKN+05b] Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On robust combiners for oblivious transfer and other primitives. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 96–113, Aarhus, Denmark, May 22–26, 2005. Springer Berlin Heidelberg, Germany.

[HR04] Chun-Yuan Hsiao and Leonid Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 92–105, Santa Barbara, CA, USA, August 15–19, 2004. Springer Berlin Heidelberg, Germany.

[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.

[JMS20] Aayush Jain, Nathan Manohar, and Amit Sahai. Combiners for functional encryption, unconditionally. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 141–168. Springer, 2020.

[Kea93] Michael Kearns. Efficient noise-tolerant learning from statistical queries. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, New York, NY, USA, 1993. Association for Computing Machinery.

[KGW+23] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 17061–17084. PMLR, 23–29 Jul 2023.

[KJGR21] Gabriel Kaptchuk, Tushar M Jois, Matthew Green, and Aviel D Rubin. Meteor: Cryptographically secure steganography for realistic distributions. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1529–1548, 2021.

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

[Lev88] Leonid A. Levin. Homogeneous measures and polynomial time invariants. In *29th Annual Symposium on Foundations of Computer Science*, pages 36–41, White Plains, NY, USA, October 24–26, 1988. IEEE Computer Society Press.

[McE78] Robert J McEliece. A public-key cryptosystem based on algebraic. *Coding Thv*, 4244(1978):114–116, 1978.

[MPW07] Remo Meier, Bartosz Przydatek, and Jürg Wullschleger. Robuster combiners for oblivious transfer. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 404–418, Amsterdam, The Netherlands, February 21–24, 2007. Springer Berlin Heidelberg, Germany.

[RBBK01] Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. In Michael K. Reiter and Pierangela Samarati, editors, *ACM CCS 2001: 8th Conference on Computer and Communications Security*, pages 196–205, Philadelphia, PA, USA, November 5–8, 2001. ACM Press.

[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.

[SBGM25] Tudor Soroceanu, Nicolas Buchmann, Theresa Graeber, and Marian Margraf. Strong multiple-CCA security in the quantum random oracle model for an FO-like PKE combiner. *IACR Communications in Cryptology*, 2(3), 2025.

[Sim98] Daniel R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 334–345, Espoo, Finland, May 31 – June 4, 1998. Springer Berlin Heidelberg, Germany.

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

[ZCW+16] Cong Zhang, David Cash, Xiuhua Wang, Xiaoqi Yu, and Sherman SM Chow. Combiners for chosen-ciphertext security. In *International Computing and Combinatorics Conference*, pages 257–268. Springer, 2016.

[ZDR19] Zachary M Ziegler, Yuntian Deng, and Alexander M Rush. Neural linguistic steganography. *arXiv preprint arXiv:1909.01496*, 2019.

Appendix

A Further Related Works

Pseudorandom codes. Pseudorandom codes (PRCs) were introduced by Christ and Gunn [CG24] and provide the possibility of robustly watermarking the output of large language models. Their constructions build on LPN-style assumptions. Since then, further constructions have been proposed by Golowich and Moitra [GM24b] based on local weak PRFs, by Christ, Golowich, Gunn, Moitra, and Wichs [CGG+25] based on permuted-code assumptions, and by Ghentiyala and Guruswami [GG25] based on the planted hyperloop assumption together with Goldreich's PRG, and based on the weak planted XOR assumption. Alrabiah et al. [AAC+25] extend a construction of a 0-bit PRC to a ideal pseudorandom code, as well as construct CCA-secure pseudorandom codes (the latter in the random oracle model). However, Döttling, Müller and Rajasree [DMSR26] as well as Garg, Gunn, and Wang [GGW25] gave strong impossibility results for PRCs from generic assumptions.

Black-box separations. Black-box separations, introduced by the seminal work by Impagliazzo and Rudich [IR89], give lower bounds on constructions of cryptographic primitives, by showing that a primitive $\mathcal{Q}$ cannot be instantiated from another primitive in a black-box way. The central idea of the result is to implement the primitive $\mathcal{P}$ by an idealized version of it, in [IR89]'s case a truly random function, which is even secure against unbounded (except in the number of queries to the random function) adversaries, while a generic adversary against any implementation of $\mathcal{Q}$ exists. This has been used in a long line of work for a variety of cryptographic primitives,

e.g. [Sim98, GT00, GKM+00, HR04, HHRS07, BPR+08, BKSY11, AS15, FR23, BMM24]. Reingold, Trevisan, and Vadhan [RTV04] as well as Baecher, Brzuska, and Fischlin [BBF13] provide a classification of different types of black-box constructions (and separations).

Lower bounds for black-box combiners. While robust black-box combiners exist for many cryptographic primitives, we also have several impossibility results for combiners: Boneh and Boyen [BB04] show that there cannot exist a black-box combiner for collision-resistant hash functions, Harnik, Kilian, Naor, Reingold, and Rosen [HKN+05b] as well as Meier, Przydatek, and Wullschleger [MPW07] show lower bounds for specific classes of combiners for oblivious transfer, and Faller, Fischlin, Hardt, and Hesse [FFHH26] show a lower bound for combining Oblivious Pseudorandom Functions.

B A Random 0-Bit PRC Oracle Secure Against Unbounded Adversaries

In this section, we will show that our random pseudorandom code construction in Definition 4.2 is indeed a PRC against unbounded adversaries. Note that the adversary has only black-box access to the PRC, the encoding rule is randomly initialized and is *not* given to the adversary as an explicit description We show that codeword length $N = \lambda^2$ already suffices to obtain a 0-bit PRC with:

- perfect robustness against a constant fraction of errors in the bounded-error model,
- perfect soundness, and
- information-theoretic pseudorandomness against *unbounded* adversaries making only polynomially many encoding queries and polynomially many challenge queries.

Note that we will analyse our construction in the bounded-error model here, in which we will have perfect robustness. In the main result of the paper, we will however use the Bernoulli-error model, which will introduce a negligible robustness error.

Parameters. Fix any constant $\tau < 1/4$ denoting the error-rate, and let $N := \lambda^2$ and $t := \tau N$. For $\ell \in \{0, \dots, N\}$, write

$$V(N, \ell) := \sum_{i=0}^{\ell} \binom{N}{i}$$

for the volume of the Hamming ball of radius ℓ in $\{0,1\}^N$. We first define the random hidden codebook.

Definition B.1. *Let $\mathcal{D}_\lambda := \{0,1\}^\lambda \times \{0,1\}^\lambda$ denote the set of labels (sk, r). Let $\mathsf{SepMap}_{\lambda,\tau}$ be the set of all maps $F : \mathcal{D}_\lambda \to \{0,1\}^N$ such that for all distinct labels $\alpha, \beta \in \mathcal{D}_\lambda$, $\Delta\big(F(\alpha), F(\beta)\big) > 2t$.*

Lemma B.2. *For every constant $\tau < 1/4$ and all sufficiently large λ, the set $\mathsf{SepMap}_{\lambda,\tau}$ is nonempty.*

Proof. The domain size is $|\mathcal{D}_\lambda| = 2^{2\lambda}$. Thus it suffices to show the existence of a binary code of blocklength $N = \lambda^2$, size $2^{2\lambda}$, and minimum distance strictly larger than $2t = 2\tau N$. The required rate is

$$R = \frac{\log_2(2^{2\lambda})}{N} = \frac{2\lambda}{\lambda^2} = \frac{2}{\lambda}.$$

Since $R \to 0$ as $\lambda \to \infty$, the Gilbert–Varshamov bound implies that for every fixed $\delta < 1/2$, and in particular for $\delta := 2\tau < 1/2$, there exists for all sufficiently large λ a binary code of rate at least R and relative distance exceeding δ. This yields a code of length $N = \lambda^2$, size $2^{2\lambda}$, and minimum distance $> 2\tau N = 2t$, as required. □

We now define the random 0-bit PRC oracle.

Definition B.3 (Random 0-bit PRC)**.** *The pair* $(\mathsf{Enc}_F, \mathsf{Dec}_F)$ *is the random* 0*-bit PRC oracle initialized as follows. Sample* $F \leftarrow \mathsf{SepMap}_{\lambda,\tau}$*. Define* $\mathsf{Enc}_F(sk; r) := F(sk, r)$ *and the decoder by*

$$\mathsf{Dec}_F(sk, y) = \top \quad \Longleftrightarrow \quad \exists r \in \{0,1\}^\lambda \text{ such that } \Delta\big(y, F(sk, r)\big) \le t.$$

Lemma B.4 (Perfect robustness and perfect soundness)**.** *For every sampled map* $F \in \mathsf{SepMap}_{\lambda,\tau}$*, the pair* $(\mathsf{Enc}_F, \mathsf{Dec}_F)$ *satisfies:*

1. Perfect robustness: *for all* sk, r, y*,*

$$\Delta\big(y, \mathsf{Enc}_F(sk; r)\big) \le t \quad \Longrightarrow \quad \mathsf{Dec}_F(sk, y) = \top.$$

2. Perfect soundness: *for every* $y \in \{0,1\}^N$*,*

$$\Pr_{sk \leftarrow \{0,1\}^\lambda}\big[\mathsf{Dec}_F(sk, y) = \top\big] \le 2^{-\lambda}.$$

Proof. Perfect robustness is immediate from the definition of Dec_F.

For perfect soundness, fix any $y \in \{0,1\}^N$. Suppose there exist distinct keys $sk \neq sk'$ such that $\mathsf{Dec}_F(sk, y) = \mathsf{Dec}_F(sk', y) = \top$. Then there exist $r, r' \in \{0,1\}^\lambda$ such that $\Delta\big(y, F(sk, r)\big) \le t$ and $\Delta\big(y, F(sk', r')\big) \le t$. By the triangle inequality, $\Delta\big(F(sk, r), F(sk', r')\big) \le 2t$, contradicting the defining property of $\mathsf{SepMap}_{\lambda,\tau}$. Thus for each y there is at most one key sk for which $\mathsf{Dec}_F(sk, y) = \top$, and the stated bound follows. □

B.1 Security experiment

We now define the pseudorandomness game. The crucial point is that the adversary gets access to the *full* encoding oracle on arbitrary inputs (sk, r), while the challenge samples are generated using a hidden challenge key.

Definition B.5 (Oracle-hidden pseudorandomness game)**.** *Let* $\mathcal{A}$ *be an adversary. The experiment* $\mathsf{Exp}_b^{\mathsf{hid\text{-}prc}}(\mathcal{A})$*, for* $b \in \{0,1\}$*, is defined as follows.*

1. *Sample* $F \leftarrow \mathsf{SepMap}_{\lambda,\tau}$*.*
2. $\mathcal{A}$ *has oracle access to the encoding oracle* $\mathcal{O}_{\mathsf{Enc}}(sk, r) := F(sk, r)$*. The adversary may ask at most* q_E *encoding queries, chosen adaptively.*
3. *Sample a hidden challenge key* $sk^\star \leftarrow \{0,1\}^\lambda$*.*
4. $\mathcal{A}$ *has oracle access to the (inputless) challenge oracle* $\mathcal{O}_{\mathsf{Chal}}^{(b)}$*, which on each invocation does the following:*

- *if $b = 1$, sample $\rho \leftarrow \{0,1\}^\lambda$ uniformly and return $\mathcal{O}^{(1)}_{\mathsf{Chal}}() := F(sk^\star, \rho)$,*
- *if $b = 0$, return $\mathcal{O}^{(0)}_{\mathsf{Chal}}() := U_N$, where U_N denotes the uniform distribution over $\{0,1\}^N$.*

The adversary may invoke the challenge oracle at most q_C times, possibly adaptively.

5. *Finally, $\mathcal{A}$ outputs a bit b'.*

We define the distinguishing advantage

$$\mathsf{Adv}^{\mathsf{hid\text{-}prc}}_{\mathsf{PRC}_0}(\mathcal{A}, \lambda) := \left| \Pr\left[\mathsf{Exp}^{\mathsf{hid\text{-}prc}}_1(\mathcal{A}) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{hid\text{-}prc}}_0(\mathcal{A}) = 1\right] \right|.$$

B.2 A convenient lazy-sampling view

For the proof it is convenient to use the following equivalent view of a random separated map. Whenever a *fresh* label is queried for the first time, its output is sampled uniformly outside the union of radius-$2t$ balls around all previously assigned outputs. Since the adversary ever sees only polynomially many labels, this lazy-sampling view completely determines the transcript distribution seen by the adversary.

More concretely, suppose the labels $\alpha_1, \ldots, \alpha_m$ are first encountered in that order during the experiment. Then their outputs are distributed as follows: $F(\alpha_1) \leftarrow U_N$ and for each $i \geq 2$,

$$F(\alpha_i) \leftarrow U\left(\{0,1\}^N \setminus \bigcup_{j<i} B(F(\alpha_j), 2t)\right).$$

This generates exactly the same ordered transcript distribution as sampling a random map $F \in \mathsf{SepMap}_{\lambda,\tau}$ and then revealing only the values on the queried labels.

Theorem B.6 (Information-theoretic security against unbounded adversaries)**.** *Fix any constant $\tau < 1/4$, and let $N = \lambda^2$ and $t = \tau N$. For any (unbounded) adversary $\mathcal{A}$ making at most q_E encoding queries and at most q_C challenge queries in the game of Definition B.5, we have*

$$\mathsf{Adv}^{\mathsf{hid\text{-}prc}}_{\mathsf{PRC}_0}(\mathcal{A}, \lambda) \leq \frac{q_E q_C}{2^{2\lambda}} + \frac{\binom{q_C}{2}}{2^\lambda} + q_C\,(q_E + q_C)\,\frac{V(N, 2t)}{2^N}.$$

In particular, if $q_E, q_C \leq \mathrm{poly}(\lambda)$, then $\mathsf{Adv}^{\mathsf{hid\text{-}prc}}_{\mathsf{PRC}_0}(\mathcal{A}, \lambda) = \mathrm{negl}(\lambda)$.

Proof. We compare the real world ($b = 1$) and the random world ($b = 0$).

Step 1: Isolate the label-collision event. Let $Q \subseteq \mathcal{D}_\lambda$ be the set of distinct labels queried by $\mathcal{A}$ to the encoding oracle, so that $|Q| \leq q_E$. In the real world, the i-th challenge query samples an independent randomness value $\rho_i \leftarrow \{0,1\}^\lambda$ and returns $F(sk^\star, \rho_i)$.

Define the bad event Bad to be the event that at least one of the following happens:

1. for some challenge query i, the label $(sk^\star, \rho_i)$ belongs to Q.
2. for some two distinct challenge queries $i \neq j$, we have $\rho_i = \rho_j$.

By the union bound, $\Pr[\mathsf{Bad}] \leq \frac{q_E q_C}{2^{2\lambda}} + \frac{\binom{q_C}{2}}{2^\lambda}$ because

- for each fixed challenge query i, the label $(sk^\star, \rho_i)$ is uniformly distributed over $\{(\mathsf{sk}^\star, \rho)\}_\rho$, hence $\Pr[(sk^\star, \rho_i) \in Q] \le \frac{|Q|}{2^\lambda} \le \frac{q_E}{2^\lambda}$
- for any fixed pair $i < j$, we have $\Pr[\rho_i = \rho_j] = 2^{-\lambda}$.

Step 2: Replace real challenges by fresh dummy labels. Condition on the event $\neg\mathsf{Bad}$. Then the challenge labels $(sk^\star, \rho_1), \ldots, (sk^\star, \rho_{q_C})$ are all fresh and pairwise distinct, and none of them belongs to Q. By the lazy-sampling view, once we condition on $\neg\mathsf{Bad}$, the ordered outputs $F(sk^\star, \rho_1), \ldots, F(sk^\star, \rho_{q_C})$ have exactly the same joint distribution as the outputs on any q_C fresh pairwise distinct dummy labels that were never queried before.

Thus, under $\neg\mathsf{Bad}$, the i-th real challenge answer is distributed as a fresh oracle value sampled uniformly from

$$\Omega_i := \{0,1\}^N \setminus \bigcup_{j=1}^{m_i} B(c_j, 2t),$$

where $c_1, c_2, \ldots$ denotes the sequence of previously assigned codewords, and before the i-th challenge answer there are at most $m_i \le q_E + i - 1 \le q_E + q_C$ previously assigned codewords.

Step 3: Compare a fresh separated sample to uniform. Fix any challenge index i. The forbidden set before answering the i-th challenge has size at most $m_i \cdot V(N, 2t) \le (q_E + q_C)\, V(N, 2t)$ by the union bound. Therefore, the statistical distance between the uniform distribution on Ω_i and the uniform distribution U_N over $\{0,1\}^N$ is at most

$$\frac{m_i\, V(N,2t)}{2^N} \le (q_E + q_C)\, \frac{V(N,2t)}{2^N}.$$

Replacing the i-th real challenge answer by a uniform random string changes the transcript distribution by at most this amount. Summing over the q_C challenge queries gives

$$q_C\, (q_E + q_C)\, \frac{V(N,2t)}{2^N}.$$

Step 4: conclude. By the triangle inequality,

$$\mathsf{Adv}^{\mathsf{hid\text{-}prc}}_{\mathsf{PRC}_0}(\mathcal{A}, \lambda) \le \Pr[\mathsf{Bad}] + q_C\, (q_E + q_C)\, \frac{V(N,2t)}{2^N}.$$

Substituting the bound on $\Pr[\mathsf{Bad}]$ from Step 1 yields

$$\mathsf{Adv}^{\mathsf{hid\text{-}prc}}_{\mathsf{PRC}_0}(\mathcal{A}, \lambda) \le \frac{q_E q_C}{2^{2\lambda}} + \frac{\binom{q_C}{2}}{2^\lambda} + q_C\, (q_E + q_C)\, \frac{V(N,2t)}{2^N},$$

as claimed. □

It remains to estimate the final term in Theorem B.6.

Lemma B.7. *For every constant $\tau < 1/4$ and $N = \lambda^2$,*

$$\frac{V(N,2t)}{2^N} = 2^{-(1 - H_2(2\tau) - o(1))N},$$

where H_2 is the binary entropy function.

Proof. Since $2t = 2\tau N$ and $2\tau < 1/2$, the standard asymptotic estimate for Hamming balls yields $V(N, 2t) = 2^{H_2(2\tau)N+o(N)}$. This implies $\frac{V(N,2t)}{2^N} = 2^{-(1-H_2(2\tau)-o(1))N}$. □

Combining Theorem B.6 with Lemma B.7, we obtain

$$\mathsf{Adv}^{\mathsf{hid\text{-}prc}}_{\mathsf{PRC}_0}(\mathcal{A}, \lambda) \leq \frac{q_E q_C}{2^{2\lambda}} + \frac{\binom{q_C}{2}}{2^\lambda} + \mathrm{poly}(\lambda) \cdot 2^{-(1-H_2(2\tau)-o(1))\lambda^2},$$

which is negligible for every fixed constant $\tau < 1/4$ and polynomially bounded q_E, q_C.

This now allows us to prove Lemma 4.3. The only point to keep in mind is that Theorem B.6 was stated for a uniformly sampled separated codebook, whereas Definition 4.2 samples the codebook entries independently. The next proof bridges the two views by conditioning on the overwhelmingly likely event that the independently sampled codebook is already separated.

Proof of Lemma 4.3. Let $N = \lambda^2$, let $t = \lfloor \tau N \rfloor$, and let $\mathcal{D}_\lambda = \{0,1\}^\lambda \times \{0,1\}^\lambda$. Write $F(\kappa, r) = C_{\kappa,r}$ for the independently sampled random map from Definition 4.2.

First, bounded-error robustness up to radius t holds for every choice of the codebook. Indeed, if $\mathsf{Ham}(y, \mathsf{Enc}(\kappa; r)) \leq t$, then the same randomness r witnesses that $\mathsf{Dec}(\kappa, y) = \top$.

Next, we show that the random codebook is separated with overwhelming probability. Let $\mathsf{Good}_{\mathsf{sep}}$ be the event that all distinct labels receive codewords at distance strictly larger than $2t$:

$$\mathsf{Good}_{\mathsf{sep}} \iff \forall \alpha \neq \beta \in \mathcal{D}_\lambda, \quad \mathsf{Ham}(F(\alpha), F(\beta)) > 2t.$$

For any fixed distinct α, β, since $F(\alpha)$ and $F(\beta)$ are independent uniform strings in $\{0,1\}^N$,

$$\Pr[\mathsf{Ham}(F(\alpha), F(\beta)) \leq 2t] = \frac{V(N, 2t)}{2^N}.$$

Therefore, by a union bound over at most $2^{4\lambda}$ ordered pairs,

$$\Pr[\neg\mathsf{Good}_{\mathsf{sep}}] \leq 2^{4\lambda} \frac{V(N, 2t)}{2^N} = 2^{4\lambda} \cdot 2^{-(1-H_2(2\tau)-o(1))N} = \mathrm{negl}(\lambda),$$

because $2\tau < 1/2$, $N = \lambda^2$, and hence $1 - H_2(2\tau) > 0$.

Conditioned on $\mathsf{Good}_{\mathsf{sep}}$, the decoder has perfect soundness. Fix any $y \in \{0,1\}^N$. If there were two distinct keys $\kappa \neq \kappa'$ such that $\mathsf{Dec}(\kappa, y) = \mathsf{Dec}(\kappa', y) = \top$, then there would exist $r, r' \in \{0,1\}^\lambda$ such that

$$\mathsf{Ham}(y, F(\kappa, r)) \leq t \quad \text{and} \quad \mathsf{Ham}(y, F(\kappa', r')) \leq t.$$

By the triangle inequality,

$$\mathsf{Ham}(F(\kappa, r), F(\kappa', r')) \leq 2t,$$

contradicting $\mathsf{Good}_{\mathsf{sep}}$. Thus, for every fixed y, at most one key κ can satisfy $\mathsf{Dec}(\kappa, y) = \top$. Hence

$$\Pr_{\kappa \leftarrow \{0,1\}^\lambda}[\mathsf{Dec}(\kappa, y) = \top] \leq 2^{-\lambda}.$$

Therefore, except with negligible probability over the initial random codebook, the decoder has perfect soundness.

It remains to prove pseudorandomness. Consider any unbounded adversary making at most q_E encoding queries and at most q_C challenge queries. Let $Q \subseteq \mathcal{D}_\lambda$ be the set of distinct encoding

labels queried by the adversary, so $|Q| \le q_E$. In the real experiment, the i-th challenge query returns $F(\mathsf{sk}^\star, \rho_i)$, where $\mathsf{sk}^\star \leftarrow \{0,1\}^\lambda$ and $\rho_i \leftarrow \{0,1\}^\lambda$.

Let Bad be the event that either some challenge label $(\mathsf{sk}^\star, \rho_i)$ belongs to Q, or two challenge randomness values collide. Since each $(\mathsf{sk}^\star, \rho_i)$ is uniform over $\mathcal{D}_\lambda$,

$$\Pr[\mathsf{Bad}] \le \frac{q_E q_C}{2^{2\lambda}} + \frac{\binom{q_C}{2}}{2^\lambda}.$$

Conditioned on $\neg\mathsf{Bad}$, all challenge labels are fresh and pairwise distinct. Since F is an independently sampled random function, the real challenge answers are independent uniform strings in $\{0,1\}^N$, independent of the adversary's encoding-query transcript. Hence, conditioned on $\neg\mathsf{Bad}$, the real and random challenge experiments have identical transcript distributions. Therefore

$$\mathsf{Adv}^{\mathsf{prc}}_{\mathsf{PRC}}(\mathcal{A}, \lambda) \le \Pr[\mathsf{Bad}] \le \frac{q_E q_C}{2^{2\lambda}} + \frac{\binom{q_C}{2}}{2^\lambda} = \mathrm{negl}(\lambda)$$

for polynomially bounded q_E, q_C.

Finally, in the Bernoulli-error model with error rate $\eta < \tau$, correctness fails only if a $\mathrm{Bin}(N, \eta)$ random variable exceeds t. By a Chernoff bound this probability is $2^{-\Omega(N)} = \mathrm{negl}(\lambda)$. Combining the bounded-error robustness, the overwhelming-probability soundness event above, and the pseudorandomness bound proves the lemma. □

C Impossibility of $(2,1)$-Combiners for IND$-CPA PKE in the Random PKE Model

In this section, we prove an impossibility result for $(2,1)$-combiners for public-key encryption with pseudorandom ciphertexts in the sparse ciphertext regime, where under any fixed public key only a negligible fraction of strings in the ambient ciphertext space are valid ciphertexts. It can be easily extended to the $(n, n-k)$ case for any $1 \le k < n$. We work in a random PKE oracle model. The goal is to show that even if the decryptor of the combiner is allowed to query all the oracles, no black-box $(2,1)$-combiner can simultaneously have overwhelming correctness and IND$-CPA security.

At a high level, the argument follows the Impagliazzo–Rudich style of proof. The adversary receives the combined public key pk. It first learns the oracle information that is likely to be shared between future honest combiner encryptions and key-generation view. Then it resamples a fake secret key sk' consistent with the learned information and with the public key pk. Finally, it receives a challenge string $\mathsf{ct}^\star$, which is either a fresh honest combiner encryption or a uniformly random string, and attempts to decrypt $\mathsf{ct}^\star$ using sk'.

The main technical point is that the decryptor of the combiner has access to all the oracles. In particular, during decryption, it may query not only the key generation and encryption oracles, but also the decryption oracles. Therefore, when we simulate the decryption oracle using a fake secret key, we must not simply answer every unknown decryption query by $\perp$. This default answer is natural in the sparse ciphertext regime, because for a fixed public key an arbitrary string is overwhelmingly likely not to be a valid ciphertext, and so its decryption should return $\perp$. But, if an earlier branch encryption query $e_j(p, \beta, r) = c$ has already been seen, then the corresponding inverse information $\mathsf{Inv}_{j,p,c} = \beta$ is forced. Hence a later branch decryption query $d_j(s, c)$, with $g_j(s) = p$, must be answered by β, not by $\perp$. Our simulation is therefore stateful and works with consistent oracle transcripts.

C.1 The random PKE oracles

We work relative to two independent labelled public-key-encryption oracles

$$O_0 = (g_0, e_0, d_0) \qquad \text{and} \qquad O_1 = (g_1, e_1, d_1).$$

For every branch $j \in \{0,1\}$, the key-generation oracle is a map $g_j : \{0,1\}^{\ell_j^{\mathsf{sk}}} \to \{0,1\}^{\ell_j^{\mathsf{pk}}}$, sampled as a uniformly random injective function where $\ell_j^{\mathsf{sk}} \le \ell_j^{\mathsf{pk}}$. The encryption oracle is a map

$$e_j : \{0,1\}^{\ell_j^{\mathsf{pk}}} \times \{0,1\} \times \{0,1\}^{\ell_j^{r}} \to \{0,1\}^{\ell_j^{\mathsf{ct}}},$$

sampled as a uniformly random injective function where $\ell_j^{\mathsf{pk}} + 1 + \ell_j^{r} \le \ell_j^{\mathsf{ct}}$. Importantly, e_j is injective as a function of the full triple (p, β, r)[8]. That is, the injectivity requirement is global:

$$(p, \beta, r) \neq (p', \beta', r') \quad \Longrightarrow \quad e_j(p, \beta, r) \neq e_j(p', \beta', r').$$

The decryption oracle is defined from g_j and e_j. For $s \in \{0,1\}^{\ell_j^{\mathsf{sk}}}$ and $c \in \{0,1\}^{\ell_j^{\mathsf{ct}}}$, define

$$d_j(s,c) = \begin{cases} \beta, & \text{if there exists } \beta \in \{0,1\}, r \in \{0,1\}^{\ell_j^{r}} \text{ such that } e_j(g_j(s), \beta, r) = c, \\ \perp, & \text{otherwise.} \end{cases}$$

This is well-defined since global injectivity of e_j implies that for fixed $(g_j(s), c)$ there is at most one pair (β, r). Thus each branch oracle is perfectly correct by construction. We assume that branch ciphertext spaces are sparse. For concreteness, it suffices to set $\ell_j^{\mathsf{ct}} = (\ell_j^{\mathsf{pk}} + \ell_j^{r} + 1)^2$ for $j \in \{0,1\}$. The exact quadratic expression is not important. What matters is that, for each fixed public key p, the number of valid ciphertexts $\{e_j(p, \beta, r) : \beta \in \{0,1\},\ r \in \{0,1\}^{\ell_j^{r}}\}$ is negligible compared to the ambient ciphertext space $\{0,1\}^{\ell_j^{\mathsf{ct}}}$.

Remark C.1. *Each single branch oracle O_j is itself IND$-CPA secure against information-theoretic adversaries in the polynomial-query regime. Indeed, for a fixed public key p and bit β, a fresh challenge ciphertext $e_j(p, \beta, r)$ is distributed like a uniformly random string in the ambient ciphertext space, except with the negligible probability that the challenge reuses one of the adversary's polynomially many previously queried randomness values.*

C.2 Definition of IND$-CPA PKE combiners

We now formally define a combiner for IND$-CPA PKE schemes.

Definition C.2 ($(n, n-k)$-combiner for IND$-CPA PKE in the random PKE model)**.** *An $(n, n-k)$-combiner for public-key encryption with pseudorandom ciphertexts in the random PKE model consists of three polynomial-query oracle algorithms*

$$\mathsf{PKE}_{\mathsf{cmb}}^{(\cdot)} = \big(\mathsf{Keygen}_{\mathsf{cmb}}^{(\cdot)}, \mathsf{Enc}_{\mathsf{cmb}}^{(\cdot)}, \mathsf{Dec}_{\mathsf{cmb}}^{(\cdot)}\big).$$

[8] This captures branch-level perfect correctness in the random-oracle model: once (p, β, r) is fixed, the resulting ciphertext determines the encrypted bit uniquely. We use this later when arguing that a previously seen encryption query fixes the corresponding inverse decryption information.

The algorithms have oracle access to component branch oracles $\mathsf{PKE}_i = (g_i, e_i, d_i)$ *for* $i \in [n]$*, where the syntaxes are*

$$g_i : \{0,1\}^{\ell_i^{\mathsf{sk}}} \to \{0,1\}^{\ell_i^{\mathsf{pk}}},$$
$$e_i : \{0,1\}^{\ell_i^{\mathsf{pk}}} \times \{0,1\} \times \{0,1\}^{\ell_i^{\rho}} \to \{0,1\}^{\ell_i^{\mathsf{ct}}},$$
$$d_i : \{0,1\}^{\ell_i^{\mathsf{sk}}} \times \{0,1\}^{\ell_i^{\mathsf{ct}}} \to \{0,1\} \cup \{\perp\}.$$

The combined algorithms have the following structure

$$\mathsf{Keygen}_{\mathsf{cmb}}^{\mathsf{PKE}_1,\ldots,\mathsf{PKE}_n} : 1^\lambda \times \{0,1\}^{\ell_{\mathsf{cmb}}^{\mathsf{kg}}} \to \{0,1\}^{\ell_{\mathsf{cmb}}^{\mathsf{pk}}} \times \{0,1\}^{\ell_{\mathsf{cmb}}^{\mathsf{sk}}},$$
$$\mathsf{Enc}_{\mathsf{cmb}}^{\mathsf{PKE}_1,\ldots,\mathsf{PKE}_n} : \{0,1\}^{\ell_{\mathsf{cmb}}^{\mathsf{pk}}} \times \{0,1\} \times \{0,1\}^{\ell_{\mathsf{cmb}}^{r}} \to \{0,1\}^{\ell_{\mathsf{cmb}}^{\mathsf{ct}}},$$
$$\mathsf{Dec}_{\mathsf{cmb}}^{\mathsf{PKE}_1,\ldots,\mathsf{PKE}_n} : \{0,1\}^{\ell_{\mathsf{cmb}}^{\mathsf{sk}}} \times \{0,1\}^{\ell_{\mathsf{cmb}}^{\mathsf{ct}}} \to \{0,1\} \cup \{\perp\}.$$

All combined length parameters $\ell_{\mathsf{cmb}}^{\mathsf{kg}}, \ell_{\mathsf{cmb}}^{\mathsf{pk}}, \ell_{\mathsf{cmb}}^{\mathsf{sk}}, \ell_{\mathsf{cmb}}^{r}, \ell_{\mathsf{cmb}}^{\mathsf{ct}}$ *are polynomially bounded in the security parameter and in the corresponding component length parameters. We require the following two properties.*

1. ***Correctness.*** *For every collection of component oracle triples* $(\mathsf{PKE}_1, \ldots, \mathsf{PKE}_n)$ *such that each* $\mathsf{PKE}_i = (g_i, e_i, d_i)$ *is a* κ_i*-correct bit-encryption public-key encryption oracle, the combined public-key encryption scheme*

$$\mathsf{PKE}_{\mathsf{cmb}}^{\mathsf{PKE}_1,\ldots,\mathsf{PKE}_n} = \left(\mathsf{Keygen}_{\mathsf{cmb}}^{\mathsf{PKE}_1,\ldots,\mathsf{PKE}_n}, \mathsf{Enc}_{\mathsf{cmb}}^{\mathsf{PKE}_1,\ldots,\mathsf{PKE}_n}, \mathsf{Dec}_{\mathsf{cmb}}^{\mathsf{PKE}_1,\ldots,\mathsf{PKE}_n}\right)$$

   is a κ_{cmb}*-correct bit-encryption public-key encryption scheme, where* κ_{cmb} *is polynomially bounded in the component correctness errors* $\kappa_1, \ldots, \kappa_n$*. The probability in the correctness guarantee is taken over the internal randomness of the combined key-generation and encryption algorithms.*

2. **IND$-CPA** ***security in the random PKE model under*** k***-branch corruption.*** *Security is defined by the following experiment with an unbounded adversary* $\mathcal{A}$ *making polynomially many oracle queries.*

   (a) *The adversary* $\mathcal{A}$*, given the public description of the combiner and all parameters, chooses a subset* $S \subseteq [n]$ *such that* $|S| \le k$*. For every* $i \in S$*, the adversary also specifies an arbitrary length-compatible stateless branch oracle* $\widetilde{\mathsf{PKE}}_i = (\widetilde{g}_i, \widetilde{e}_i, \widetilde{d}_i)$*. Equivalently, the adversary fixes the complete truth tables of the corrupted branch oracles at this security parameter. The corrupted branches need not satisfy security.*

   (b) *The challenger sets* $\mathsf{PKE}_i := \widetilde{\mathsf{PKE}}_i$ *for every* $i \in S$*. For every honest index* $i \in [n] \setminus S$*, the challenger independently samples* $\mathsf{PKE}_i = (g_i, e_i, d_i)$ *as a random PKE oracle with the corresponding branch parameters.*

   (c) *The challenger samples* $(\mathsf{pk}^\star, \mathsf{sk}^\star) \leftarrow \mathsf{Keygen}_{\mathsf{cmb}}^{\mathsf{PKE}_1,\ldots,\mathsf{PKE}_n}(1^\lambda)$*. The adversary receives* $\mathsf{pk}^\star$*.*

   (d) *The challenger samples a hidden experiment bit*

$$\sigma \leftarrow \{\text{real}, \text{rand}\}.$$

   The adversary is given adaptive oracle access to the component branch oracles and the challenge oracle. These queries may be interleaved arbitrarily.

i. ***Component-oracle access.*** *For every $i \in [n]$, the adversary may query*

$$\mathcal{O}_g(i, s) := g_i(s),$$

$$\mathcal{O}_e(i, p, \beta, r) := e_i(p, \beta, r),$$

and

$$\mathcal{O}_d(i, s, c) := d_i(s, c).$$

Thus the adversary has oracle access to the full branch oracles, including branch decryption oracles.
Since the combined key-generation and encryption algorithms are public oracle algorithms, the adversary may also run them on inputs and randomness of its choice using its component-oracle access.

ii. ***Challenge oracle.*** *The adversary may query a chosen message bit $\beta \in \{0,1\}$ to the challenge oracle $\mathcal{C}_\sigma$. If $\sigma = \text{real}$, the oracle samples $\omega \leftarrow \{0,1\}^{\ell^r_{\mathsf{cmb}}}$ and returns $\mathsf{Enc}^{\mathsf{PKE}_1,\ldots,\mathsf{PKE}_n}_{\mathsf{cmb}}(\mathsf{pk}^\star, \beta; \omega)$. If $\sigma = \text{rand}$, the oracle returns an independent uniform string $U_{\ell^{\mathsf{ct}}_{\mathsf{cmb}}} \leftarrow \{0,1\}^{\ell^{\mathsf{ct}}_{\mathsf{cmb}}}$.*

(e) Finally, $\mathcal{A}$ outputs a bit b', where $b' = 1$ means "real" and $b' = 0$ means "random".

The advantage of $\mathcal{A}$ is

$$\mathsf{Adv}^{\mathit{IND\$\text{-}CPA}}_{\mathcal{A}} := \left|\Pr[b' = 1 : \sigma = \text{real}] - \Pr[b' = 1 : \sigma = \text{rand}]\right|,$$

where the probability is over the random choice of the honest random PKE branch oracles, the combined key-generation randomness, the fresh randomness used by the challenge oracle, and the internal randomness of $\mathcal{A}$.

The combiner is secure if, for every unbounded adversary $\mathcal{A}$ making polynomially many oracle queries, there exists a negligible function $\text{negl}(\cdot)$ such that

$$\mathsf{Adv}^{\mathit{IND\$\text{-}CPA}}_{\mathcal{A}} \leq \text{negl}(\lambda).$$

C.3 Cells and closed transcripts

The oracles g_j, e_j, d_j are correlated rather than independent. For example, once an encryption query $e_j(p, \beta, r) = c$ has been observed, the corresponding inverse information $\mathsf{Inv}_{j,p,c} = \beta$ is also determined. We record this information as an inverse cell, rather than writing $d_j(g_j^{-1}(p), c) = \beta$, because p need not be in the image of g_j. In particular, $g_j^{-1}(p)$ may not exist.

Similarly, a decryption query $d_j(s, c)$ reveals more than just the decrypted value. It first fixes the public key $p = g_j(s)$, and then reveals the inverse value of c relative to this public key. Thus, if $d_j(s, c) = \mu$, the closed transcript should contain both $G_{j,s} = p$ and $\mathsf{Inv}_{j,p,c} = \mu$, where $\mu \in \{0, 1, \perp\}$.

To keep track of these correlated information, we represent oracle information by branch-indexed cells using $G_{j,s}, E_{j,p,\beta,r}$ and $\mathsf{Inv}_{j,p,c}$. Their intended values are

$$G_{j,s} = g_j(s), \qquad E_{j,p,\beta,r} = e_j(p, \beta, r), \qquad \mathsf{Inv}_{j,p,c} \in \{0, 1, \perp\}.$$

For a labelled oracle view V, let $\mathsf{Cells}(V)$ denote the set of cell-value pairs revealed by V, closed under the following rules.

1. If V contains a query-answer pair $g_j(s) = p$, then $\mathsf{Cells}(V)$ contains $G_{j,s} = p$.

2. If V contains a query-answer pair $e_j(p, \beta, r) = c$, then $\mathsf{Cells}(V)$ contains both $E_{j,p,\beta,r} = c$ and $\mathsf{Inv}_{j,p,c} = \beta$.

3. If V contains a query-answer pair $d_j(s, c) = \mu$, then the answer is determined by the underlying public key $p = g_j(s)$ and the inverse cell $\mathsf{Inv}_{j,p,c}$. In the closed transcript, we therefore record the unique value $p = g_j(s)$ together with the corresponding inverse information. Thus $\mathsf{Cells}(V)$ contains $G_{j,s} = p$ and $\mathsf{Inv}_{j,p,c} = \mu$.

We write $\mathrm{clr}(P)$ for the closure of a set P of cell-value pairs under these rules. Whenever we maintain a transcript, we always maintain it in closed form.

For a set P of cell-value pairs, let $\mathsf{Loc}(P)$ denote the corresponding set of cell locations, obtained by forgetting the assigned values. Thus, for example,

$$\mathsf{Loc}\big(\{G_{j,s} = p,\ E_{j,p',\beta,r} = c,\ \mathsf{Inv}_{j,p'',c'} = \mu\}\big) = \{G_{j,s},\ E_{j,p',\beta,r},\ \mathsf{Inv}_{j,p'',c'}\}.$$

For a view V, we abbreviate $\mathsf{Loc}(V) := \mathsf{Loc}(\mathsf{Cells}(V))$. A closed transcript is said to be locally consistent if it satisfies all finite constraints forced by the random PKE model. In particular:

- no cell location is assigned two different values;
- the assigned G-cells are consistent with injectivity of g_j;
- the assigned E-cells are consistent with injectivity of e_j;
- whenever $E_{j,p,\beta,r} = c$ appears, the transcript also contains $\mathsf{Inv}_{j,p,c} = \beta$;
- no transcript contains both $\mathsf{Inv}_{j,p,c} = \perp$ and $\mathsf{Inv}_{j,p,c} = \beta$ for $\beta \in \{0, 1\}$.
- no transcript contains both $\mathsf{Inv}_{j,p,c} = \perp$ and $E_{j,p,\beta,r} = c$ for some β, r.

C.4 Notations for the impossibility proof

As mentioned in the beginning of this section, we will focus on the $(2, 1)$-case. Let $\mathsf{PKE}_{\mathsf{cmb}} = \big(\mathsf{Keygen}_{\mathsf{cmb}}^{O_0,O_1}, \mathsf{Enc}_{\mathsf{cmb}}^{O_0,O_1}, \mathsf{Dec}_{\mathsf{cmb}}^{O_0,O_1}\big)$ be a polynomial-query bit-encryption combiner using the two random PKE branches O_0, O_1. The decryptor is allowed to query all the oracles, i.e., for each $j \in \{0, 1\}$, it may query g_j, e_j, and d_j.

Let $N = N(\lambda)$ be an upper bound on the number of labelled oracle queries made by each combiner algorithm. Let κ be an absolute constant such that any view containing at most N labelled oracle queries reveals at most κN cells after closure.

Let $\rho(\lambda)$ denote the correctness probability of the combiner: for every fixed pair of branch PKE oracles (O_0, O_1) and every $b \in \{0, 1\}$,

$$\Pr[\mathsf{Dec}_{\mathsf{cmb}}^{O_0,O_1}(\mathsf{sk}, \mathsf{ct}) = b : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Keygen}_{\mathsf{cmb}}^{O_0,O_1}(1^\lambda), \mathsf{ct} \leftarrow \mathsf{Enc}_{\mathsf{cmb}}^{O_0,O_1}(\mathsf{pk}, b)] \geq \rho(\lambda),$$

where the probability is over the randomness of the combined key-generation, encryption, and decryption algorithms.

The adversary constructed below is computationally unbounded but makes only polynomially many oracle queries.

Theorem C.3 (IR-style impossibility for $(2,1)$-IND\$-CPA PKE combiners with full branch-oracle decryptors)**.** *Let* $\mathsf{PKE}_{\mathsf{cmb}} = \left(\mathsf{Keygen}_{\mathsf{cmb}}^{O_0,O_1}, \mathsf{Enc}_{\mathsf{cmb}}^{O_0,O_1}, \mathsf{Dec}_{\mathsf{cmb}}^{O_0,O_1}\right)$ *be any polynomial-query black-box* $(2,1)$*-combiner for bit-encryption IND\$-CPA* PKE *in the random PKE branch oracle model with sparse ciphertext spaces, i.e.,*

$$\ell_j^{\mathsf{ct}} - \ell_j^{r} = \omega(\log(\lambda)) \qquad \text{for each } j \in \{0,1\}.$$

Then no such combiner can be IND\$-CPA secure with correctness $\rho(\lambda) \geq 1 - \mathrm{negl}(\lambda)$*.*

The remainder of this section proves Theorem C.3.

C.5 The eavesdropper

We now describe the adversary Eve. Fix $\delta(\lambda) = 1/\mathrm{poly}(\lambda)$ for some large polynomial $\mathrm{poly}(\lambda)$. Set $\tau := \frac{\delta(\lambda)}{2\kappa N}$. Let $\gamma := c \cdot \lceil \frac{\kappa N}{\delta(\lambda)} \log\left(\frac{\kappa N}{\delta(\lambda)}\right) \rceil$, where $c > 0$ is a sufficiently large universal constant. Eve receives the combined public key pk. Her preprocessing phase does not use the challenge oracle $\mathcal{C}_\sigma$. Since encryption is public, she can run honest combined encryptions herself under the public key pk.

Eve initializes $I := \emptyset$. For every $i \in [\gamma]$ and $\beta \in \{0,1\}$, Eve samples fresh encryption randomness and runs $\mathsf{Enc}_{\mathsf{cmb}}^{O_0,O_1}(\mathsf{pk}, \beta)$. She records the full labelled encryption view and adds the resulting cell-value pairs to I.

Thus, if a self-generated encryption view contains a branch key-generation query $g_j(s) = p$, then Eve records $G_{j,s} = p$. If it is a branch encryption query $e_j(p, \alpha, r) = c$, then Eve records both $E_{j,p,\alpha,r} = c$ and $\mathsf{Inv}_{j,p,c} = \alpha$. If the view contains a branch decryption query $d_j(s, c) = \mu$, then Eve also learns $p = g_j(s)$, if this value was not already known, and records $G_{j,s} = p$ and $\mathsf{Inv}_{j,p,c} = \mu$.

After the learning phase, Eve samples a fake key-generation view $K' \leftarrow \mathsf{Samp}_{\mathsf{Keygen}}(\mathsf{pk}, I)$. We now explain this notation carefully. Consider an honest execution of $\mathsf{Keygen}_{\mathsf{cmb}}^{O_0,O_1}(1^\lambda)$. Such an execution samples internal randomness for $\mathsf{Keygen}_{\mathsf{cmb}}$, makes some oracle queries to O_0, O_1, receives answers, and eventually outputs a pair $(\mathsf{pk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{out}})$.

The complete information consisting of the internal randomness, all oracle queries and answers, and the final output is called the key-generation view. We write it as $K = \mathsf{View}_{\mathsf{Keygen}_{\mathsf{cmb}}}^{O_0,O_1}(1^\lambda)$. The view K determines a set of revealed cells $\mathsf{Cells}(K)$, and it also determines the output pair $\mathsf{out}(K) = (\mathsf{pk}_{\mathsf{out}}, \mathsf{sk}_{\mathsf{out}})$.

The sampler $\mathsf{Samp}_{\mathsf{Keygen}}(\mathsf{pk}, I)$ denotes the exact conditional distribution of such honest key-generation views conditioned on two events:

1. the key-generation execution outputs public key pk, and
2. the revealed cells in the key-generation view are consistent with the already learned transcript I.

More explicitly,

$$\mathsf{Samp}_{\mathsf{Keygen}}(\mathsf{pk}, I) := \left(K \;\middle|\; \begin{array}{l} O_0', O_1' \text{ are sampled from the} \\ \quad \text{random PKE-oracle distribution} \\ K = \mathsf{View}_{\mathsf{Keygen}_{\mathsf{cmb}}}^{O_0',O_1'}(1^\lambda), \\ \mathsf{out}_{\mathsf{pk}}(K) = \mathsf{pk}, \\ \mathsf{Cells}(K) \cup I \text{ is locally consistent} \end{array} \right).$$

Here $\mathsf{out}_{\mathsf{pk}}(K)$ denotes the public-key component of the output of the view K. Local consistency means that no cell is assigned two different values, that the already revealed G- and E-cells respect the injectivity constraints of g_j and e_j, and that the inverse cells are consistent with the encryption cells.

Equivalently, Eve is resampling a possible secret key sk' and a possible key-generation transcript that could have produced the already observed public key pk, without contradicting any oracle information that Eve has already learned during preprocessing. Importantly, Eve does not require this fake key-generation view to be the original one used to generate pk. It only needs to be distributed exactly as an honest key-generation view conditioned on the information available to Eve.

Since Eve is computationally unbounded, she can implement this conditional sampling exactly. For example, she may enumerate all possible internal randomness strings for $\mathsf{Keygen}_{\mathsf{cmb}}$, all possible oracle-answer patterns consistent with I and with the random labelled-PKE oracle distribution, keep precisely those views that output pk, and sample from the resulting conditional distribution with the correct probabilities. The efficiency of this sampling is irrelevant for the separation argument; only the number of actual oracle queries made by Eve is required to be polynomial.

Let sk' be the secret key contained in the fake key-generation view K'. Eve defines the initial closed transcript $P_0 := \mathrm{clr}\big(I \cup \mathsf{Cells}(K')\big)$. Only after this preprocessing does Eve query the IND\$-CPA challenge oracle. She samples $b^\star \leftarrow \{0,1\}$ and queries $\mathsf{ct}^\star \leftarrow \mathcal{C}_\sigma(b^\star)$. In the real experiment, $\mathsf{ct}^\star = \mathsf{Enc}_{\mathsf{cmb}}^{O_0,O_1}(\mathsf{pk}, b^\star)$ with fresh independent randomness. Let $B^\star$ denote the hidden honest encryption view producing $\mathsf{ct}^\star$. In the random experiment, $\mathsf{ct}^\star = U_{\ell_{\mathsf{cmb}}^{\mathsf{ct}}}$.

Eve now runs $\mathsf{Dec}_{\mathsf{cmb}}^{(\cdot,\cdot)}(\mathsf{sk}', \mathsf{ct}^\star)$ using a mixed, stateful simulation of the branch oracles. The simulation maintains a dynamically growing closed transcript P_t, initialized as $P_t := P_0$. Every answer returned to the decryptor is added to P_t, together with all cell-value pairs forced by closure.

Simulating g_j-queries. For a query $g_j(s)$, Eve answers as follows:

$$\widetilde{g}_{P_t,j}(s) = \begin{cases} P_t(G_{j,s}), & \text{if } G_{j,s} \in \mathsf{Loc}(P_t), \\ g_j(s), & \text{otherwise.} \end{cases}$$

In the second case, Eve queries the real g_j-oracle, obtains $p = g_j(s)$, adds $G_{j,s} = p$ to P_t, and returns p.

Simulating e_j-queries. For a query $e_j(p, \beta, r)$, Eve answers as follows:

$$\widetilde{e}_{P_t,j}(p,\beta,r) = \begin{cases} P_t(E_{j,p,\beta,r}), & \text{if } E_{j,p,\beta,r} \in \mathsf{Loc}(P_t), \\ e_j(p,\beta,r), & \text{otherwise.} \end{cases}$$

In the second case, Eve queries the real e_j-oracle, obtains $c = e_j(p,\beta,r)$, adds $E_{j,p,\beta,r} = c$ and $\mathsf{Inv}_{j,p,c} = \beta$ to P_t, and returns c.

Simulating d_j-queries. For a query $d_j(s,c)$, Eve first computes the mixed public key $p := \widetilde{g}_{P_t,j}(s)$. She then answers by inverse-cell lookup:

$$\widetilde{d}_{P_t,j}(s,c) = \begin{cases} P_t(\mathsf{Inv}_{j,p,c}), & \text{if } \mathsf{Inv}_{j,p,c} \in \mathsf{Loc}(P_t), \\ \bot, & \text{otherwise.} \end{cases}$$

If $\mathsf{Inv}_{j,p,c} \notin \mathsf{Loc}(P_t)$, Eve adds $\mathsf{Inv}_{j,p,c} = \bot$ to P_t and returns $\bot$.

This last rule is the key point of the combined simulation. Eve does not answer all unknown-looking branch decryption queries by $\bot$ blindly. She first checks whether the corresponding inverse cell is already known. If, earlier in the same decryptor execution or during preprocessing, an encryption query $e_j(p, \beta, r) = c$ has been observed, then closure has already inserted $\mathsf{Inv}_{j,p,c} = \beta$ into P_t. Therefore a later query $d_j(s, c)$ with $\widetilde{g}_{P_t,j}(s) = p$ is answered by β, not by $\bot$.

Remark C.4. *One might try to define the simulation differently: whenever $\mathsf{Inv}_{j,p,c} \notin P_t$, Eve could query the real branch decryption oracle $d_j(s, c)$ and return its answer. This, however, does not remove the main difficulty, because the real oracle d_j is tied to the real value $g_j(s)$, whereas the simulated execution is tied to $p = \widetilde{g}_{P_t,j}(s)$. For example, the fake key-generation view may contain $G_{j,s'} = p$, where p is the public key used by an honest encryption to produce c. In the simulated world, s' is treated as the secret key for p. But in the real oracle world the preimage of p may be some $s \neq s'$, so $g_j(s) = p$ while $g_j(s') \neq p$. Hence the real decryption relevant to p is $d_j(s, c)$, whereas querying $d_j(s', c)$ decrypts relative to the different public key $g_j(s')$. Thus $d_j(s', c)$ may differ from $d_j(s, c)$, and the correct simulated answer must be determined by $\mathsf{Inv}_{j,p,c}$, not by the real value $d_j(s', c)$.*

We therefore deliberately do not query the real decryption oracle on unseen inverse cells. Our goal is to compile out previously unseen branch-decryption information: if the inverse cell $\mathsf{Inv}_{j,p,c}$ has not already been forced by the transcript, then the simulator treats the ciphertext as invalid and answers $\bot$. The proof below shows that this replacement is correct except with negligible probability, because a fresh unknown branch ciphertext is valid only with negligible probability in the sparse random labelled-PKE model. This is the direct analogue of the "answer $\bot$ on unseen decode queries" strategy used in the PRC-combiner impossibility proof (see Section 7).

The algorithm in Figure 1 summarizes Eve.

C.6 Learning encryption/key-generation intersections

The first step is to show that Eve's preprocessing learns all relevant intersections between the fake key-generation view and the challenge-encryption view. Define, in the real experiment,

$$\mathsf{Good}_{\mathsf{Enc}} \quad \Longleftrightarrow \quad \mathsf{Loc}(K') \cap \mathsf{Loc}(B^\star) \subseteq \mathsf{Loc}(I).$$

Thus $\mathsf{Good}_{\mathsf{Enc}}$ says that every cell shared by the fake key-generation view and the hidden challenge-encryption view was already learned during Eve's preprocessing phase.

Lemma C.5 (Encryption/key-generation intersections are learned)**.** *In the real IND$-CPA experiment, after Eve's learning phase and challenge ciphertext generation,*

$$\Pr[\mathsf{Good}_{\mathsf{Enc}}] \geq 1 - \delta(\lambda),$$

where the probability is over all randomness of the experiment.

Proof. Fix the sampled branch oracles O_0, O_1, the public key pk, and a bit $\beta \in \{0, 1\}$. For every closed branch-indexed cell location a, define

$$p_\beta(a) := \Pr_r \left[a \in \mathsf{Loc}\big(\mathsf{View}^{O_0,O_1}_{\mathsf{Enc}_{\mathsf{cmb}}}(\mathsf{pk}, \beta; r)\big) \right],$$

Algorithm 1 Adversary $\mathsf{Eve}^{O_0,O_1,\mathcal{C}_\sigma}(\mathsf{pk})$

Input: combined public key pk, branch oracles O_0, O_1, and challenge oracle $\mathcal{C}_\sigma$
Output: a bit in $\{0,1\}$

Preprocessing phase.

$I \leftarrow \emptyset$
for $i \in [\gamma]$ and $\beta \in \{0,1\}$ **do**
    Run $\mathsf{Enc}_{\mathsf{cmb}}^{O_0,O_1}(\mathsf{pk}, \beta)$ with fresh randomness and add all revealed cell-value pairs to I
end for
$K' \leftarrow\$ \mathsf{Samp}_{\mathsf{Keygen}}(\mathsf{pk}, I)$
$\mathsf{sk}' \leftarrow$ the secret-key component of $\mathsf{out}(K')$
$P_0 \leftarrow \mathrm{clr}(I \cup \mathsf{Cells}(K'))$

Decryption phase.

Sample $b^\star \leftarrow\$ \{0,1\}$ and query $\mathsf{ct}^\star \leftarrow \mathcal{C}_\sigma(b^\star)$
Run $\mathsf{Dec}_{\mathsf{cmb}}^{(\cdot,\cdot)}(\mathsf{sk}', \mathsf{ct}^\star)$ with the mixed stateful branch-oracle simulation described above, initialized at P_0
Let D' be the resulting full decryptor view and let P_{fin} be the final closed transcript
if $\mathsf{out}(D') \in \{0,1\}$ **then**
    $\widehat{b} \leftarrow \mathsf{out}(D')$
else
    $\widehat{b} \leftarrow\$ \{0,1\}$
end if
if $\widehat{b} = b^\star$ **then**
    return 1
end if
return 0

Figure 1: Algorithm Eve

where the probability is over the fresh randomness r used by the combined encryption algorithm. Since every encryption view contains at most N labelled oracle queries, and hence at most κN cells after closure, we have $\sum_a p_\beta(a) \le \kappa N$. Call a cell location a β-heavy if $p_\beta(a) \ge \tau$. The number of β-heavy cells is at most $\frac{\kappa N}{\tau}$.

During preprocessing, Eve performs γ independent self-generated encryptions of the bit β. Therefore the probability that a fixed β-heavy cell is missed in all γ samples is at most $(1-\tau)^\gamma \le e^{-\gamma\tau}$. By a union bound over all β-heavy cells, the probability that some β-heavy cell is missed is at most $\frac{\kappa N}{\tau} e^{-\gamma\tau}$. Taking a union bound over both values $\beta \in \{0,1\}$, and using the choice of γ and τ, this probability is at most $\frac{\delta(\lambda)}{2}$.

Condition on the event that no heavy encryption cell was missed. After I and K' are fixed, define $T := \mathsf{Loc}(K') \setminus \mathsf{Loc}(I)$. Since K' is a key-generation view containing at most N labelled oracle queries, $|T| \le \kappa N$. Moreover, every cell in T is $b^\star$-light. Indeed, if a cell in T were $b^\star$-heavy, then it would have been learned during the preprocessing encryptions of $b^\star$, and hence would already lie in I.

Now consider the fresh challenge-encryption view $B^\star$. By a union bound,

$$\begin{aligned}\Pr[\mathsf{Loc}(B^\star) \cap T \neq \emptyset] &\le \sum_{a \in T} p_{b^\star}(a) \\ &< |T|\tau \\ &\le \kappa N \tau \\ &= \frac{\delta(\lambda)}{2}.\end{aligned}$$

Combining the probability of missing a heavy cell with the above light-cell intersection bound gives $\Pr[\neg \mathsf{Good}_{\mathsf{Enc}}] \le \delta(\lambda)$. This proves the lemma. □

C.7 The only possible decryptor-simulation error

We next isolate the only way in which Eve's mixed decryptor simulation can differ from a genuine decryptor execution in a completed branch-oracle world. The point of this subsection is to separate two different issues.

First, there is the issue of whether the responses that Eve gives to the oracle queries made during $\mathsf{Dec}_{\mathsf{cmb}}^{(\cdot,\cdot)}(\mathsf{sk}', \mathsf{ct}^*)$ are correct. This is the simulation issue. Second, there is the issue of whether the finite transcript accumulated so far can be embedded into a globally valid pair of branch oracles satisfying the injectivity constraints of g_j and e_j. This is the consistency issue, captured by the event Cons defined at the end of this subsection.

C.7.1 Forward queries and local consistency

There are only three kinds of branch-oracle queries that the combiner decryptor can make: g_j-queries, e_j-queries, and d_j-queries. We begin with the forward queries.

Consider first a query $g_j(s)$. If the cell $G_{j,s}$ already belongs to the current closed transcript $\mathsf{Loc}(P_t)$, then Eve answers with the stored value $P_t(G_{j,s})$. Any full oracle completion extending P_t must assign exactly this value to $g_j(s)$, so this answer is correct relative to every such completion. If $G_{j,s} \notin \mathsf{Loc}(P_t)$, Eve queries the real branch oracle g_j, obtains a value $p = g_j(s)$, adds the cell $G_{j,s} = p$ to P_t, and returns p. Thus a fresh g_j-query also creates no simulation error: Eve does not

guess its answer. She obtains the actual value and records it. The only possible problem would be that this new value accidentally violates injectivity with some previously fixed fake G-cell, but this is not a decryptor-simulation error; it is precisely a local-consistency failure and is included in the later event $\neg\mathsf{Cons}$.

Now consider a query $e_j(p, \beta, r)$. If the cell $E_{j,p,\beta,r}$ already belongs to $\mathsf{Loc}(P_t)$, then Eve returns the stored value $P_t(E_{j,p,\beta,r})$. Again, any oracle completion extending P_t must give exactly this answer. If $E_{j,p,\beta,r} \notin \mathsf{Loc}(P_t)$, Eve queries the real encryption oracle e_j, obtains $c = e_j(p, \beta, r)$, adds both $E_{j,p,\beta,r} = c$ and $\mathsf{Inv}_{j,p,c} = \beta$ to P_t, and returns c. Hence a fresh e_j-query also creates no simulation error. Eve again does not guess the answer; she learns the actual value of the forward encryption cell and also records the inverse cell forced by this encryption query. As before, if this newly learned encryption value accidentally collides with some already fixed encryption value, or contradicts an already fixed inverse cell, then the total transcript is not locally consistent. Such a failure is accounted for by $\neg\mathsf{Cons}$, not by the event defined in this subsection.

Therefore, neither g_j-queries nor e_j-queries are a source of simulation error. The reason is simple: when the relevant cell is already known, Eve returns the known value. When it is not known, Eve queries the real forward oracle and then records the exact answer. Thus forward-oracle queries are always answered exactly, up to the separate issue of whether the growing finite transcript remains globally consistent.

C.7.2 The only possible simulation error: unseen inverse cells

The only delicate case is a branch decryption query. Consider a query $d_j(s, c)$ during the mixed decryptor execution. Eve first computes the public key $p := \widetilde{g}_{P_t,j}(s)$. By the discussion above, this computation itself does not introduce a simulation error: either $G_{j,s}$ was already in $\mathsf{Loc}(P_t)$, in which case every completion extending P_t has $g_j(s) = P_t(G_{j,s})$, or Eve has just queried the real g_j-oracle and added $G_{j,s} = p$ to P_t.

Having determined p, the correct branch decryption answer is governed by the inverse cell $\mathsf{Inv}_{j,p,c}$. If $\mathsf{Inv}_{j,p,c} \in \mathsf{Loc}(P_t)$, then Eve answers with the stored value $P_t(\mathsf{Inv}_{j,p,c})$. This again creates no new simulation error: any full oracle completion extending P_t must assign this exact value to the inverse cell. This includes all three possibilities. If the stored value is 0 or 1, then Eve returns that bit. If the stored value is $\perp$, then Eve returns $\perp$. In particular, if an earlier encryption query $e_j(p, \beta, r) = c$ was observed, then closure has already inserted $\mathsf{Inv}_{j,p,c} = \beta$ into P_t, and Eve correctly answers the decryption query by β, not by $\perp$.

Thus the only remaining case is the following: the decryptor asks $d_j(s, c)$, Eve computes $p := \widetilde{g}_{P_t,j}(s)$, and the inverse cell $\mathsf{Inv}_{j,p,c}$ is not present in $\mathsf{Loc}(P_t)$. In this case Eve answers $\perp$ and adds $\mathsf{Inv}_{j,p,c} = \perp$ to the transcript. This is the only place where Eve may be making a guess. The guess is valid if, in the full branch oracle, the ciphertext c is not a valid encryption under public key p. It is invalid exactly if there exist $\beta \in \{0,1\}$ and $r \in \{0,1\}^{\ell_j^r}$ such that $e_j(p, \beta, r) = c$. Equivalently, the true value of the inverse cell is a bit, $\mathsf{Inv}_{j,p,c} \in \{0,1\}$, but Eve has just answered $\perp$.

This motivates the following definition. For an input $x \in \{0,1\}^{\ell^{\mathsf{ct}}_{\mathsf{cmb}}}$, define $\mathsf{Bad}_{\mathsf{Dec}}(x)$ to be the event that, during the mixed decryptor execution on input x, some branch decryption query $d_j(s, c)$ is asked such that, with $p := \widetilde{g}_{P_t,j}(s)$, the inverse cell $\mathsf{Inv}_{j,p,c}$ is not in $\mathsf{Loc}(P_t)$ at the time of the query, but the true inverse value in the original PKE oracle is non-$\perp$.

Let $\mathsf{Good}_{\mathsf{Dec}}(x) := \neg\mathsf{Bad}_{\mathsf{Dec}}(x)$. We next define the final transcript and the consistency event.

C.7.3 The final transcript and the consistency event

Let $P_{\mathsf{fin}}(x)$ denote the final closed transcript obtained at the end of the mixed decryptor execution on input x. We will also use the following consistency event. For an honest encryption view B, define $T_{\mathsf{fin}}(x;B) := \mathrm{clr}\big(I \cup \mathsf{Cells}(K') \cup \mathsf{Cells}(B) \cup P_{\mathsf{fin}}(x)\big)$, where in the random-input experiment, we take $B = \emptyset$. Let $\mathsf{Cons}(x;B)$ be the event that $T_{\mathsf{fin}}(x;B)$ is completable to a valid pair of branch PKE oracles. That is, no cell location is assigned two different values, the assigned G-cells and E-cells are compatible with the injectivity of the corresponding g_j and e_j, and all assigned inverse cells are compatible with the assigned forward encryption cells.

We write $\mathsf{Cons}_{\mathrm{rand}} := \mathsf{Cons}(U;\emptyset)$ for the random-input experiment, and $\mathsf{Cons}_{\mathrm{real}} := \mathsf{Cons}(\mathsf{ct}^\star;B^\star)$ for the real-encryption experiment, where $B^\star$ is the honest encryption view producing $\mathsf{ct}^\star$.

On $\mathsf{Good}_{\mathsf{Dec}}(x)$, every decryption query that Eve answers by lookup is correct because the relevant inverse cell is already in P_t, and every decryption query whose inverse cell is not yet in P_t is correctly answered by $\perp$. As explained above, all g_j-queries and e_j-queries are answered exactly by construction. Therefore, on $\mathsf{Good}_{\mathsf{Dec}}(x) \wedge \mathsf{Cons}(x;B)$, the entire mixed decryptor view is identical to a genuine decryptor view in some valid completed branch-oracle world extending the final closed transcript $T_{\mathsf{fin}}(x;B)$.

Lemma C.6 (Only unseen inverse cells can cause simulation error)**.** *For every input $x \in \{0,1\}^{\ell_{\mathsf{ct}}}$ and every honest encryption view B, if $\mathsf{Good}_{\mathsf{Dec}}(x) \wedge \mathsf{Cons}(x;B)$ holds, then the mixed decryptor execution on input x is identical to a genuine execution of*

$$\mathsf{Dec}_{\mathsf{cmb}}^{\widehat{O}_0,\widehat{O}_1}(\mathsf{sk}',x)$$

in some valid completion $(\widehat{O}_0,\widehat{O}_1)$ of $T_{\mathsf{fin}}(x;B)$.

Proof. The discussion above exhausts the three kinds of branch-oracle queries made by the decryptor. For g_j-queries and e_j-queries, the simulator either returns a value already fixed in the current closed transcript or queries the real forward oracle and records the exact answer. Hence these query types never introduce a simulation error.

For a d_j-query, if the relevant inverse cell $\mathsf{Inv}_{j,p,c}$ is already present in P_t, then the simulator again returns the uniquely forced value. Thus the only possible error is the unknown-inverse-cell case encoded by $\mathsf{Bad}_{\mathsf{Dec}}(x)$. On $\mathsf{Good}_{\mathsf{Dec}}(x)$, every such unseen inverse cell is truly $\perp$, so these answers are also correct.

Finally, $\mathsf{Cons}(x;B)$ guarantees that the final transcript $T_{\mathsf{fin}}(x;B)$ extends to valid branch PKE oracles. Since every simulated answer agrees with that completion, the entire mixed decryptor execution is identical to a genuine decryptor view in some valid completion of $T_{\mathsf{fin}}(x;B)$. □

C.8 Sparse inverse cells for random combiner inputs

We first show that $\mathsf{Bad}_{\mathsf{Dec}}(x)$ is very unlikely when the decryptor input is a uniformly random combiner-length string, except on the separate consistency failure event. This is the analogue of the sparse acceptance statement in the random PRC model (see Section 5).

Lemma C.7 (Random-input decryptor reliability)**.** *There exist negligible functions $\mu_0(\lambda)$ and $\mu_1(\lambda)$ such that*

$$\Pr[\mathsf{Bad}_{\mathsf{Dec}}(U) \wedge \mathsf{Cons}_{\mathrm{rand}}] \le N \cdot (\mu_0(\lambda) + \mu_1(\lambda)),$$

where the probability is over the random choice of (O_0, O_1)*, the uniform input* $U \leftarrow \{0,1\}^{\ell^{\mathsf{ct}}_{\mathsf{cmb}}}$*, and the randomness used to generate* I, K', sk'*, and* P_0*. More explicitly, if the total transcript size is bounded by a polynomial* $Q(\lambda)$*, then*

$$\mu_j(\lambda) \leq \frac{2^{\ell^r_j+1}}{2^{\ell^{\mathsf{ct}}_j} - Q(\lambda)} \leq 2^{\ell^r_j+2-\ell^{\mathsf{ct}}_j}$$

for all sufficiently large λ*. Consequently, if* $\Pr[\neg\mathsf{Cons}_{\mathrm{rand}}] \leq \mathrm{negl}(\lambda)$*, then*

$$\Pr[\mathsf{Bad}_{\mathsf{Dec}}(U)] \leq N \cdot (\mu_0(\lambda) + \mu_1(\lambda)) + \mathrm{negl}(\lambda).$$

Proof. We bound the probability that a bad unknown inverse cell is encountered while the final transcript remains consistent. Suppose that $\mathsf{Bad}_{\mathsf{Dec}}(U) \wedge \mathsf{Cons}_{\mathrm{rand}}$ occurs, and consider the first branch decryption query at which $\mathsf{Bad}_{\mathsf{Dec}}(U)$ occurs. Let this query be $d_j(s,c)$, and let $p := \widetilde{g}_{P_t,j}(s)$ be the public key computed by the mixed simulation immediately before answering the query.

Since $\mathsf{Cons}_{\mathrm{rand}}$ holds, the current prefix transcript is completable to valid branch PKE oracles. If $\mathsf{Inv}_{j,p,c} \in \mathsf{Loc}(P_t)$, then Eve answers using the stored inverse value, and this query cannot cause $\mathsf{Bad}_{\mathsf{Dec}}(U)$. Therefore, at the first bad query we must have $\mathsf{Inv}_{j,p,c} \notin \mathsf{Loc}(P_t)$. In this case Eve is about to answer $\perp$. This answer is wrong exactly if c is nevertheless a valid encryption under public key p, namely if there exist $\beta \in \{0,1\}$ and $r \in \{0,1\}^{\ell^r_j}$ such that $e_j(p,\beta,r) = c$.

We now condition on the entire previous transcript up to this point, including U, the fake key sk', the current closed transcript P_t, and all previous oracle answers and simulated answers. Under this conditioning, the query (j,s,c) and the value p are fixed. Moreover, because $\mathsf{Cons}_{\mathrm{rand}}$ holds, the conditioned prefix is completable. Since the inverse cell $\mathsf{Inv}_{j,p,c}$ is not yet in $\mathsf{Loc}(P_t)$, the event that it is non-$\perp$ can only happen if the still-unrevealed image of one of the at most $2 \cdot 2^{\ell^r_j}$ inputs

$$(p,\beta,r) \in \{p\} \times \{0,1\} \times \{0,1\}^{\ell^r_j}$$

hits the fixed ciphertext c.

Because e_j is a uniformly random injective function, and because at most $Q(\lambda)$ ciphertext values have already been fixed in the transcript, this conditional probability is at most

$$\mu_j(\lambda) \leq \frac{2^{\ell^r_j+1}}{2^{\ell^{\mathsf{ct}}_j} - Q(\lambda)}.$$

This bound is valid despite the adaptivity of the decryptor queries, since it is taken after conditioning on the complete previous transcript. If c had already been produced by an earlier encryption query with the same public key p, then closure would already have inserted the corresponding inverse cell into P_t, contrary to the unknown-cell case. If c had already been produced by an earlier encryption query at a different input, then injectivity of e_j, together with consistency of the prefix, prevents it from also being the image of some (p,β,r).

The decryptor makes at most N labelled oracle queries, hence at most N branch decryption queries. Taking a union bound over all possible decryption query times and over the two branches gives

$$\Pr[\mathsf{Bad}_{\mathsf{Dec}}(U) \wedge \mathsf{Cons}_{\mathrm{rand}}] \leq N \cdot (\mu_0(\lambda) + \mu_1(\lambda)).$$

Finally, $\mu_j(\lambda) \leq 2^{\ell^r_j+2-\ell^{\mathsf{ct}}_j}$ follows from $Q(\lambda) \leq 2^{\ell^{\mathsf{ct}}_j-1}$ for all sufficiently large λ. Under the sparse ciphertext-space condition, this quantity is negligible.

The final unconditional statement follows immediately from

$$\Pr[\mathsf{Bad}_{\mathsf{Dec}}(U)] \le \Pr[\mathsf{Bad}_{\mathsf{Dec}}(U) \wedge \mathsf{Cons}_{\mathrm{rand}}] + \Pr[\neg \mathsf{Cons}_{\mathrm{rand}}].$$

□

C.9 Gap implies break: transferring from random strings to real encryptions

We now transfer the random-input reliability bound to real combiner ciphertexts. This is the standard pseudorandomness-transfer step: if the bad inverse-cell event occurred noticeably more often on real combiner encryptions than on uniformly random strings, then this event would distinguish real combiner ciphertexts from random strings.

Lemma C.8 (Transfer from random strings to real encryptions)**.** *Assume that the combiner is IND$-CPA secure against adversaries that may corrupt either one of the two branch oracles. Then, in the real experiment, where* $\mathsf{ct}^\star = \mathsf{Enc}_{\mathsf{cmb}}^{O_0,O_1}(\mathsf{pk}, b^\star)$*,*

$$\Pr[\mathsf{Bad}_{\mathsf{Dec}}(\mathsf{ct}^\star)] \le N \cdot (\mu_0(\lambda) + \mu_1(\lambda)) + \mathrm{negl}(\lambda).$$

Proof. We first make the event branch-wise. For each $j \in \{0,1\}$, let $\mathsf{Bad}_{\mathsf{Dec},j}(x)$ denote the event that, during the mixed decryptor execution on input x, the bad unknown inverse-cell event occurs at a branch-j decryption query. More explicitly, $\mathsf{Bad}_{\mathsf{Dec},j}(x)$ occurs if the simulated decryptor asks some query $d_j(s,c)$, Eve computes $p := \widetilde{g}_{P_t,j}(s)$, the inverse cell $\mathsf{Inv}_{j,p,c}$ is not yet in $\mathsf{Loc}(P_t)$, Eve therefore answers $\perp$, but the true value of $\mathsf{Inv}_{j,p,c}$ in the original branch oracle is a bit.

Thus $\mathsf{Bad}_{\mathsf{Dec}}(x) = \mathsf{Bad}_{\mathsf{Dec},0}(x) \cup \mathsf{Bad}_{\mathsf{Dec},1}(x)$. We prove that, for each fixed $j \in \{0,1\}$, $\Pr[\mathsf{Bad}_{\mathsf{Dec},j}(\mathsf{ct}^\star)] \le \Pr[\mathsf{Bad}_{\mathsf{Dec},j}(U)] + \mathrm{negl}(\lambda)$, where $U \leftarrow \{0,1\}^{\ell_{\mathsf{cmb}}^{\mathsf{ct}}}$. Summing this inequality over the two branches and then using the branch-wise form of Lemma C.7 gives the lemma.

Fix a branch $j \in \{0,1\}$. Suppose, toward a contradiction, that for some inverse-polynomial $\alpha(\lambda) > 0$, we have $\Pr[\mathsf{Bad}_{\mathsf{Dec},j}(\mathsf{ct}^\star)] \ge \Pr[\mathsf{Bad}_{\mathsf{Dec},j}(U)] + \alpha(\lambda)$. We construct a IND$-CPA adversary $\mathcal{A}_j$ that corrupts branch j and distinguishes real combiner ciphertexts from random strings with inverse-polynomial advantage.

The adversary $\mathcal{A}_j$ works in the IND$-CPA experiment for the combiner. It receives the combined public key pk and oracle access to the challenge oracle $\mathcal{C}_\sigma$, where $\sigma \in \{\mathrm{real}, \mathrm{rand}\}$. It corrupts branch j, and hence knows the full branch oracle $O_j = (g_j, e_j, d_j)$. In particular, for any public key p and ciphertext c, it can decide exactly whether the inverse cell $\mathsf{Inv}_{j,p,c}$ is $\perp$ or a bit, by checking whether there exist $\beta \in \{0,1\}$ and $r \in \{0,1\}^{\ell_j^r}$ such that $e_j(p,\beta,r) = c$.

Description of $\mathcal{A}_j$. First, $\mathcal{A}_j$ performs exactly the same preprocessing as Eve. This preprocessing is independent of the challenge oracle. Namely, $\mathcal{A}_j$ runs the honest combined encryption algorithm under pk many times, records the full labelled oracle views, closes these views into cells, and obtains the learned closed transcript I. Then it samples $K' \leftarrow \mathsf{Samp}_{\mathsf{Keygen}}(\mathsf{pk}, I)$ from the exact posterior distribution of key-generation views that output pk and are consistent with I. Let sk' be the secret key contained in K', and let $P_0 := \mathrm{clr}(I \cup \mathsf{Cells}(K'))$.

Since this preprocessing does not use $\mathcal{C}_\sigma$, the joint distribution of $\mathsf{pk}, I, K', P_0, \mathsf{sk}'$ is the same whether $\mathcal{C}_\sigma$ is the real encryption oracle or the random-string oracle.

Next, $\mathcal{A}_j$ samples $b^\star \leftarrow \{0,1\}$, queries $\mathsf{ct}^\star \leftarrow \mathcal{C}_\sigma(b^\star)$, and runs the same mixed stateful decryptor simulation on input $\mathsf{ct}^\star$ that Eve would run, initialized with transcript P_0. The only extra action is that $\mathcal{A}_j$ monitors whether $\mathsf{Bad}_{\mathsf{Dec},j}(\mathsf{ct}^\star)$ occurs.

During the simulated decryptor execution, $\mathcal{A}_j$ answers all g- and e-queries exactly as Eve does. If the queried cell is already in the current transcript P_t, it returns the stored value. If a fresh $g_{j'}$- or $e_{j'}$-query is asked, it queries the corresponding branch oracle and records the answer, together with all cells forced by closure.

Now consider a branch decryption query $d_{j'}(s, c)$. The simulation first computes $p := \widetilde{g}_{P_t,j'}(s)$. If $\mathsf{Inv}_{j',p,c} \in \mathsf{Loc}(P_t)$, the simulator answers using the stored inverse value. This cannot cause a new bad event.

The only interesting case is when $\mathsf{Inv}_{j',p,c} \notin \mathsf{Loc}(P_t)$, so the mixed simulation is about to answer $\perp$ and insert $\mathsf{Inv}_{j',p,c} = \perp$ into P_t. If $j' \neq j$, then $\mathcal{A}_j$ does not try to determine whether this was a bad event, since it has corrupted only branch j. It continues exactly as Eve would.

If $j' = j$, then $\mathcal{A}_j$ can test exactly whether the answer $\perp$ is wrong. Since it knows the full map e_j, it checks whether there exist $\beta \in \{0,1\}$ and $r \in \{0,1\}^{\ell_j^r}$ such that $e_j(p, \beta, r) = c$. If such a pair exists, then the true inverse value is $\mathsf{Inv}_{j,p,c} = \beta$, and the mixed simulation is about to answer $\perp$ incorrectly; in this case $\mathcal{A}_j$ sets a flag $\mathsf{bad} = 1$. If no such pair exists, then the true inverse value is $\perp$. In both cases, the simulation itself continues exactly as Eve's simulation would.

At the end of the simulated decryptor execution, $\mathcal{A}_j$ outputs 1 if and only if the flag bad was set.

Analysis. For every fixed preprocessing transcript $\mathsf{pk}, I, K', P_0, \mathsf{sk}'$, the procedure above computes exactly the predicate $x \mapsto \mathsf{Bad}_{\mathsf{Dec},j}(x)$. The only nontrivial part of evaluating this predicate is deciding whether an unknown branch-j inverse cell $\mathsf{Inv}_{j,p,c}$ is actually non-$\perp$, and corruption of branch j gives $\mathcal{A}_j$ exactly this information.

Therefore, when $\mathcal{C}_\sigma = \mathcal{C}_{\mathrm{real}}$, the challenge string is a fresh real combiner ciphertext, so $\Pr[\mathcal{A}_j^{\mathcal{C}_{\mathrm{real}}}(\mathsf{pk}) = 1] = \Pr[\mathsf{Bad}_{\mathsf{Dec},j}(\mathsf{ct}^\star)]$. When $\mathcal{C}_\sigma = \mathcal{C}_{\mathrm{rand}}$, the challenge string is a uniformly random string U, so $\Pr[\mathcal{A}_j^{\mathcal{C}_{\mathrm{rand}}}(\mathsf{pk}) = 1] = \Pr[\mathsf{Bad}_{\mathsf{Dec},j}(U)]$. By the assumed gap, $\mathcal{A}_j$ distinguishes real combiner ciphertexts from uniformly random strings with advantage at least $\alpha(\lambda)$, while corrupting only branch j. This contradicts $\mathsf{IND\$\text{-}CPA}$ security under single-branch corruption.

Hence, for every $j \in \{0,1\}$, we have $\Pr[\mathsf{Bad}_{\mathsf{Dec},j}(\mathsf{ct}^\star)] \leq \Pr[\mathsf{Bad}_{\mathsf{Dec},j}(U)] + \mathrm{negl}(\lambda)$. Taking a union bound over the two branches,

$$\Pr[\mathsf{Bad}_{\mathsf{Dec}}(\mathsf{ct}^\star)] \leq \sum_{j \in \{0,1\}} \Pr[\mathsf{Bad}_{\mathsf{Dec},j}(U)] + \mathrm{negl}(\lambda).$$

Finally, the proof of Lemma C.7 gives the branch-wise bound $\Pr[\mathsf{Bad}_{\mathsf{Dec},j}(U) \wedge \mathsf{Cons}_{\mathrm{rand}}] \leq N \cdot \mu_j(\lambda)$. Since $\Pr[\neg\mathsf{Cons}_{\mathrm{rand}}] \leq \mathrm{negl}(\lambda)$, summing over $j \in \{0,1\}$ yields

$$\Pr[\mathsf{Bad}_{\mathsf{Dec}}(\mathsf{ct}^\star)] \leq N \cdot (\mu_0(\lambda) + \mu_1(\lambda)) + \mathrm{negl}(\lambda).$$

This proves the lemma. □

C.10 Fake-key resampling preserves correctness

We now show that Eve's fake key sk' preserves correctness except when one of the bad events above occurs. The only delicate point is that the fake key-generation view must be sampled from the correct posterior distribution. We make this explicit first.

Posterior interpretation of the fake key-generation sampler. Recall that $K' \leftarrow \mathsf{Samp}_{\mathsf{Keygen}}(\mathsf{pk}, I)$ means that K' is sampled from the exact posterior distribution of an honest key-generation view conditioned on the public key pk and the learned transcript I. In particular, K' is weighted according to its probability under the random PKE-oracle distribution, subject to $\mathsf{out}_{\mathsf{pk}}(K') = \mathsf{pk}$ and $\mathsf{Cells}(K') \cup I$ being locally consistent. Since Eve is computationally unbounded, she can implement this sampler exactly by enumerating finite key-generation oracle stubs consistent with (pk, I) and sampling them with their correct posterior weights. We use the following posterior-switching fact.

Lemma C.9 (Posterior switching for key-generation views)**.** *Fix the public key* pk *and the learned closed transcript* I*. Let* K *be the original key-generation view that produced* pk*, and let* $K' \leftarrow \mathsf{Samp}_{\mathsf{Keygen}}(\mathsf{pk}, I)$*. Then, conditioned on* (pk, I)*, the finite views* K *and* K' *have the same distribution as oracle stubs.*

Moreover, let B *be a fresh honest encryption view under* pk*. On the event that* $\mathsf{Loc}(\mathsf{Cells}(K')) \cap \mathsf{Loc}(\mathsf{Cells}(B)) \subseteq \mathsf{Loc}(I)$ *and* $I \cup \mathsf{Cells}(K') \cup \mathsf{Cells}(B)$ *is completable, the joint finite transcript* $I \cup \mathsf{Cells}(K') \cup \mathsf{Cells}(B)$ *has the same posterior distribution as the corresponding honest transcript* $I \cup \mathsf{Cells}(K) \cup \mathsf{Cells}(B)$*, where* K *is the original key-generation view.*

Proof. The first statement is exactly the definition of $\mathsf{Samp}_{\mathsf{Keygen}}(\mathsf{pk}, I)$: it samples a key-generation view from the posterior distribution of the original key-generation view conditioned on (pk, I).

For the second statement, condition on fixed finite stubs for I, K', and B. If K' and B reveal no common cell location outside I, then all cell values revealed by $K' \setminus I$ and $B \setminus I$ are sampled from disjoint unrevealed parts of the random branch-oracle distribution, subject only to the finite injectivity and inverse-cell constraints. Hence their joint probability factors exactly as in an honest execution. If the combined finite transcript is completable, the posterior probability assigned to this finite stub is precisely the probability that the random PKE-oracle distribution assigns to the same stub in the honest experiment. Thus replacing the original key-generation view K by an independent posterior sample K' does not change the joint distribution of the finite transcript, except on the excluded events where K' and B have an unknown intersection or where the finite stub is not completable. □

We next record the consistency bound needed in the real-encryption experiment. Recall that $\mathsf{Cons}_{\mathrm{real}} = \mathsf{Cons}(\mathsf{ct}^\star; B^\star)$, where $B^\star$ is the honest encryption view producing the challenge ciphertext $\mathsf{ct}^\star$.

Lemma C.10 (Completing the finite transcript)**.** *In the real experiment,*

$$\Pr[\neg \mathsf{Cons}_{\mathrm{real}} \wedge \mathsf{Good}_{\mathsf{Enc}}] \leq \mathrm{negl}(\lambda).$$

Equivalently, conditioned on $\mathsf{Good}_{\mathsf{Enc}}$*, the event* $\neg\mathsf{Cons}_{\mathrm{real}}$ *occurs only with negligible probability.*

Proof. On $\mathsf{Good}_{\mathsf{Enc}}$, every cell location shared by the fake key-generation view K' and the honest encryption view $B^\star$ is already contained in the learned transcript I. Thus K' and $B^\star$ create no unknown intersection. The remaining transcript $T_{\mathsf{fin}}(\mathsf{ct}^\star; B^\star)$ contains only polynomially many cell assignments, because Eve performs polynomially many preprocessing encryptions and the combiner algorithms make at most $N(\lambda)$ oracle queries per execution.

A failure of $\mathsf{Cons}_{\mathrm{real}}$ can therefore occur only if one of the polynomially many newly sampled forward values accidentally collides with, or contradicts, a previously fixed value. This includes a fresh $g_j(s)$-answer colliding with an already fixed $G_{j,s'}$-value for $s' \neq s$, a fresh $e_j(p, \beta, r)$-answer

colliding with an already fixed encryption value for a different input triple, or a fresh encryption value contradicting an already fixed inverse cell. Each such event has probability inverse-exponentially small in the relevant output length, because g_j and e_j are length-expanding random injective functions. A union bound over polynomially many possible collisions gives the claimed negligible bound. □

We can now prove the fake-key correctness lemma.

Lemma C.11 (Fake-key resampling with full branch-oracle decryptor)**.** *Assume that the combiner is IND$-CPA secure under single-branch corruption. Then, in the real experiment,*

$$\Pr[\widehat{b} = b^\star] \geq \rho(\lambda) - \delta(\lambda) - N \cdot (\mu_0(\lambda) + \mu_1(\lambda)) - \mathrm{negl}(\lambda).$$

Proof. In the real experiment, let $\mathsf{ct}^\star = \mathsf{Enc}_{\mathsf{cmb}}^{O_0,O_1}(\mathsf{pk}, b^\star)$, and let $B^\star$ be the hidden honest encryption view producing $\mathsf{ct}^\star$. By Lemma C.5, $\Pr[\neg\mathsf{Good}_{\mathsf{Enc}}] \leq \delta(\lambda)$. By Lemma C.8,

$$\Pr[\mathsf{Bad}_{\mathsf{Dec}}(\mathsf{ct}^\star)] \leq N \cdot (\mu_0(\lambda) + \mu_1(\lambda)) + \mathrm{negl}(\lambda).$$

By Lemma C.10, $\Pr[\neg\mathsf{Cons}_{\mathrm{real}} \wedge \mathsf{Good}_{\mathsf{Enc}}] \leq \mathrm{negl}(\lambda)$. Define the good event

$$\mathcal{G} := \mathsf{Good}_{\mathsf{Enc}} \wedge \mathsf{Good}_{\mathsf{Dec}}(\mathsf{ct}^\star) \wedge \mathsf{Cons}_{\mathrm{real}}.$$

We first argue that, on $\mathcal{G}$, Eve's fake execution is a genuine correctness experiment with a posterior-resampled key-generation view.

Because $\mathsf{Good}_{\mathsf{Enc}}$ holds, every cell location shared by K' and $B^\star$ is already contained in I. Therefore K' and $B^\star$ have no unknown intersection. Because $\mathsf{Cons}_{\mathrm{real}}$ holds, the closed transcript $T_{\mathsf{fin}}(\mathsf{ct}^\star; B^\star)$ is completable to full valid branch PKE oracles $\widehat{O}_0 = (\widehat{g}_0, \widehat{e}_0, \widehat{d}_0)$ and $\widehat{O}_1 = (\widehat{g}_1, \widehat{e}_1, \widehat{d}_1)$. In such a completion, K' is a valid key-generation view producing $(\mathsf{pk}, \mathsf{sk}')$, and $B^\star$ is a valid encryption view producing $\mathsf{ct}^\star = \mathsf{Enc}_{\mathsf{cmb}}^{\widehat{O}_0,\widehat{O}_1}(\mathsf{pk}, b^\star)$.

By Lemma C.6, applied with $x = \mathsf{ct}^\star$ and $B = B^\star$, the mixed decryptor view is identical to a genuine decryptor execution in this completed branch-oracle world. Consequently,

$$D' = \mathsf{View}_{\mathsf{Dec}_{\mathsf{cmb}}}^{\widehat{O}_0,\widehat{O}_1}(\mathsf{sk}', \mathsf{ct}^\star).$$

Now we use Lemma C.9. Since $K' \leftarrow \mathsf{Samp}_{\mathsf{Keygen}}(\mathsf{pk}, I)$ is an exact posterior sample of the key-generation view, and since $\mathsf{Good}_{\mathsf{Enc}}$ rules out all unknown intersections between K' and $B^\star$, the finite transcript $I \cup \mathsf{Cells}(K') \cup \mathsf{Cells}(B^\star)$ has the same distribution, on the good event, as the corresponding honest transcript using the original key-generation view K. Thus the fake-key experiment on $\mathcal{G}$ is distributed exactly as an honest correctness experiment, except that we have discarded the bad events $\neg\mathsf{Good}_{\mathsf{Enc}}$, $\mathsf{Bad}_{\mathsf{Dec}}(\mathsf{ct}^\star)$, and $\neg\mathsf{Cons}_{\mathrm{real}} \wedge \mathsf{Good}_{\mathsf{Enc}}$.

By correctness of the combiner, in the honest correctness experiment we have $\Pr[\mathsf{Dec}_{\mathsf{cmb}}^{O_0,O_1}(\mathsf{sk}, \mathsf{ct}) = b] \geq \rho(\lambda)$. Therefore,

$$\begin{aligned}\Pr[\mathsf{out}(D') = b^\star] &\geq \rho(\lambda) - \Pr[\neg\mathsf{Good}_{\mathsf{Enc}}] - \Pr[\mathsf{Bad}_{\mathsf{Dec}}(\mathsf{ct}^\star)] - \Pr[\neg\mathsf{Cons}_{\mathrm{real}} \wedge \mathsf{Good}_{\mathsf{Enc}}] \\ &\geq \rho(\lambda) - \delta(\lambda) - N \cdot (\mu_0(\lambda) + \mu_1(\lambda)) - \mathrm{negl}(\lambda).\end{aligned}$$

Finally, Eve sets $\widehat{b} = \mathsf{out}(D')$ whenever $\mathsf{out}(D') \in \{0, 1\}$, and otherwise sets $\widehat{b}$ to a uniformly random bit. Hence $\Pr[\widehat{b} = b^\star] \geq \Pr[\mathsf{out}(D') = b^\star]$, which proves the lemma. □

C.11 Combining the distinguishing attacks

Let $\mathcal{E}$ denote Eve's final distinguisher, which outputs 1 if and only if $\widehat{b} = b^\star$.

Lemma C.12 (Random challenge gives no information about $b^\star$)**.** *In the random experiment,*

$$\Pr[\mathcal{E}^{\mathcal{C}_{\mathrm{rand}}}(\mathsf{pk}) = 1] = \frac{1}{2}.$$

Proof. In the random experiment,

$$\mathsf{ct}^\star = \mathcal{C}_{\mathrm{rand}}(b^\star) = U_{\ell^{\mathsf{ct}}_{\mathsf{cmb}}}.$$

The challenge string $\mathsf{ct}^\star$ is independent of $b^\star$. Eve's preprocessing is also independent of $b^\star$, and the mixed decryptor simulation takes as input only $\mathsf{ct}^\star$ together with the preprocessing transcript. Hence the output bit $\widehat{b}$ is independent of $b^\star$, so $\Pr[\widehat{b} = b^\star] = 1/2$. Since $\mathcal{E}$ outputs 1 exactly when $\widehat{b} = b^\star$, the lemma follows. □

Proof of Theorem C.3. Let $\mathcal{E}$ be Eve. Recall that $\mathcal{E}$ outputs 1 if and only if $\widehat{b} = b^\star$. First consider the real experiment. In this case, $\mathsf{ct}^\star = \mathcal{C}_{\mathrm{real}}(b^\star) = \mathsf{Enc}^{O_0,O_1}_{\mathsf{cmb}}(\mathsf{pk}, b^\star)$. By Lemma C.11,

$$\Pr[\mathcal{E}^{\mathcal{C}_{\mathrm{real}}}(\mathsf{pk}) = 1] = \Pr[\widehat{b} = b^\star] \geq \rho(\lambda) - \delta(\lambda) - N \cdot (\mu_0(\lambda) + \mu_1(\lambda)) - \mathrm{negl}(\lambda).$$

By Lemma C.12,

$$\begin{aligned}\mathsf{Adv}_{\mathsf{IND\$\text{-}CPA}}(\mathcal{E}) &= \left|\Pr[\mathcal{E}^{\mathcal{C}_{\mathrm{real}}}(\mathsf{pk}) = 1] - \Pr[\mathcal{E}^{\mathcal{C}_{\mathrm{rand}}}(\mathsf{pk}) = 1]\right| \\ &\geq \rho(\lambda) - \delta(\lambda) - N \cdot (\mu_0(\lambda) + \mu_1(\lambda)) - \frac{1}{2} - \mathrm{negl}(\lambda).\end{aligned}$$

By the branch ciphertext sparsity assumption, each $\mu_j(\lambda)$ is negligible. Since $N(\lambda)$ is polynomial, $N \cdot (\mu_0(\lambda) + \mu_1(\lambda)) = \mathrm{negl}(\lambda)$. Thus, if $\rho(\lambda) \geq 1 - \mathrm{negl}(\lambda)$ and $\delta(\lambda) = 1/\mathrm{poly}(\lambda)$, then

$$\mathsf{Adv}_{\mathsf{IND\$\text{-}CPA}}(\mathcal{E}) \geq \frac{1}{2} - \frac{1}{\mathrm{poly}(\lambda)},$$

which is non-negligible. This contradicts IND$-CPA security.

Therefore no polynomial-query $(2, 1)$-combiner for IND$-CPA PKE in this random PKE model can have overwhelming correctness and IND$-CPA security. □