Ideal Pseudorandom Code, Revisited

Ganyuan Cao[1]

Télécom Paris, Institut Polytechnique de Paris, Palaiseau, France
ganyuan.cao@telecom-paris.fr

Abstract. Pseudorandom error-correcting codes (PRCs), introduced by Christ and Gunn at CRYPTO'24, combine pseudorandomness with error correction, providing a natural abstraction for robust watermarking and steganography on generative AI models. Subsequent standalone notions, which are ideal security for secret-key PRCs and CCA-style security for public-key PRCs, are oracle-based and do not capture composable use with explicit parties, sessions, and corruption.

We give a UC treatment of PRCs via corruption-aware ideal functionalities for both settings. Under non-adaptive corruption, the UC notions recover the standalone ones. Under adaptive corruption, we identify a common obstruction: dummy codewords sampled before corruption must later be opened as valid PRC codewords. We formalize this as a decoder-non-committing code (NC-PRC), which any adaptively UC-secure realization must induce and which conversely suffices for one. We then capture failures of such openings via targeted low opening capacity, show that it rules out NC-PRCs, and prove that LDPC-based PRCs have this property and hence do not admit an NC-PRC.

On the positive side, we sketch two heuristic compilers to lift error-correcting codes to admit NC-PRCs: a secret-key one from a puncturable PRF and indistinguishability obfuscation (iO), and a public-key one from a smooth projective hash function (SPHF), both evading the barrier via programmable acceptance.

Finally, we identify a fresh-codeword explanation barrier for public-key PRCs: accepted unseen codewords cannot be explained from public information without violating pseudorandomness, hence public-key UC realizations require a trapdoor or an idealized setup, and PRC-based watermark detection is inherently designated-verifier.

Keywords: Pseudorandom Error-Correcting Code (PRC) · Universal Composability (UC) · Non-Committing PRC

1 Introduction

Pseudorandom (error-correcting) codes (PRCs), first introduced by Christ and Gunn [CG24], combine two properties: codewords are computationally indistinguishable from uniform strings, yet remain decodable after bounded errors. This makes PRCs a natural abstraction for robust watermarking and steganography [GZS24, CHS25, CG24], especially on large language models (LLMs), where such watermarks can provide cryptographic evidence that content was generated by

a watermarked model, a principled tool for provenance, plagiarism, and misuse detection. The original construction [CG24] relies on low-density parity-check (LDPC) codes and LPN-type assumptions [BFKL94], and later works explore alternatives such as planted hyperloops [GG24], weak PRFs [GM24], and permuted puzzles [CGG+25]. Recent black-box separations [GGW25, DMR25] also suggest this reliance on coding-theoretic structure is inherent and PRCs cannot be built from "classic" cryptographic primitives in a black-box way. Complementing these, Francati et al. [FGP+26] establish an unconditional limit on robustness, where no such code tolerates a corruption rate above $1 - 1/q$ over an alphabet of size q. More recently, Döttling et al. [DKRR26] show that black-box robust combiners, which merge several candidates into one that remains secure as long as one of them is, exist neither for secret-key PRCs, nor for public-key encryption with pseudorandom ciphertexts over sparse ciphertext spaces, which is a primitive closely related to public-key PRCs.

Alrabiah et al. [AAC+25] revisited PRCs through stronger standalone security notions. For secret-key PRCs, they introduced an "ideal security" experiment capturing pseudorandom encodings, robust decoding of words near honestly generated codewords, and rejection of words unrelated to the encoding transcript. For public-key PRCs, they defined a chosen-codeword security notion, analogous to IND-CCA security for public-key encryption but adapted to robustness balls around challenge codewords. They also gave a Fujisaki–Okamoto-style [FO99] compiler from suitably robust and pseudorandom PRCs to these stronger notions. More recent works [DJK+26, CMY26] have also explored constructions of CCA(1)-secure PRCs in the standard model from other assumptions.

Despite these advances, existing notions still fall short of how PRCs are beginning to be used. Recent works already deploy PRCs as components inside larger cryptographic systems, for example in the unforgeable watermark construction of Lin et al. [LSS26], and the mismatch sharpens once a PRC is composed with primitives whose standard treatment is already a UC ideal functionality, such as oblivious transfer (OT) or vector oblivious linear evaluation (VOLE), since a standalone guarantee cannot be plugged into the composition theorem. Oracle-based games leave unspecified how a PRC behaves under explicit party corruption, and how valid codewords generated outside the prescribed oracle interface should be handled. This motivates us to study under what conditions PRCs admit UC-secure realizations, where security must hold under explicit parties, sessions, corruptions, and concurrent composition.

1.1 Our Contribution

We revisit the ideal and CCA security framework for pseudorandom codes (PRCs) of [CG24, AAC+25] from the perspective of composability. Our main message is that the standalone oracle notions of [AAC+25] have a natural UC interpretation under non-adaptive corruption, but adaptive corruption introduces new opening and explanation requirements that are absent from the standalone games.

UC Functionality and Non-Adaptive Equivalence. In Section 3, we define a UC ideal functionality $\mathcal{F}_{\mathsf{PRC}}$ for both secret-key and public-key PRCs (Fig. 2). The functionality records honestly issued codewords in a table, and answers decoding queries using the same table-first rule as the ideal and CCA experiments of [AAC+25]. The new feature is corruption awareness, that is, depending on the corruption status of $\mathcal{P}_{\mathsf{ecd}}$ and $\mathcal{P}_{\mathsf{dcd}}$, fresh codewords outside the table are either rejected or delegated to the ideal-process adversary.

In Section 4, we show that, under non-adaptive corruption, $\mathcal{F}_{\mathsf{PRC}}$ recovers the standalone notions of [AAC+25]. In the secret-key setting it corresponds to the ideal security experiment, while in the public-key setting it corresponds to the random branch of the IND-CCA experiment. Thus, for non-adaptive corruptions, UC security of the natural protocol is equivalent to the corresponding standalone oracle notion (Theorems 1 and 2).

Theorems 1 and 2 (Informal). *For a natural PRC protocol, UC security against non-adaptive corruption is equivalent to ideal security in the secret-key setting, and to IND-CCA security in the public-key setting.*

Adaptive Corruption and Decoder Openings. In Section 5, we show that the above equivalence breaks under adaptive decoder corruption. In the ideal execution, honest codewords are sampled as dummy uniform strings by $\mathcal{F}_{\mathsf{PRC}}$. After corrupting the decoder, however, the simulator must reveal a decoder state $\mathsf{dk}^\star$ under which these dummy strings look like honestly generated PRC codewords. This resembles the opening requirement in non-committing encryption [CFGN96, DN00, CHK05, CLNS17], but the PRC setting imposes a code-specific condition. Once a dummy codeword c is opened to a message m, the opened decoder state must decode c to m and, in the secret-key case with $\mathsf{ek} = \mathsf{dk}$, where corrupting the encoder leaks dk while the decoder remains honest, also the relevant δ-neighborhood of c, so that the opening is not merely pointwise. We formalize this requirement as the existence of a *decoder-non-committing pseudorandom code* (NC-PRC, Definition 9). We prove that adaptive UC security with decoder-state exposure is equivalent to the standalone notion together with such an object (Theorem 3). In the public-key case, its opening game also lets the adversary decode fresh codewords before corruption, since those decoded with the real key must decode in the same way under the opened state.

Theorem 3 (Informal). *A natural PRC protocol is UC-secure against adaptive decoder corruption if and only if the PRC satisfies ideal security (secret-key) or IND-CCA security (public-key) and admits a decoder-non-committing pseudorandom code.*

A Structural Gap for Security against Adaptive Corruption. To identify when such decoder openings are impossible, we introduce *targeted low opening capacity* (TLOC, Definition 13). This property captures a structural limitation of the decoder family: after many independent uniform strings have been fixed, no syntactically valid decoder state can post-hoc explain them as codewords

for prescribed messages. We show that targeted low opening capacity rules out decoder-non-committing pseudorandom codes, and therefore rules out adaptive UC security with decoder-state exposure (Lemma 1).

Lemma 1 (Informal). *Targeted low opening capacity (TLOC) rules out the existence of decoder-non-committing pseudorandom codes.*

Targeted low opening capacity is a strictly stronger demand than the (random-string form of) soundness (Definition 14) naturally relevant for PRCs. The latter only requires that a uniformly sampled word be rejected under an honest decoder state generated *independently* of it, whereas the former quantifies over *every* syntactically valid decoder state, including those chosen adversarially after the dummy codewords are fixed. Soundness therefore constrains a single honestly generated state, while targeted low opening capacity constrains the entire decoder family.

This, in turn, suggests a design criterion for UC-secure PRCs against adaptive corruption, that is, a candidate code should remain sound under honest keys, while avoiding targeted low opening capacity so that the ideal-process adversary still has enough structured opening capacity to explain uniform dummy codewords after corruption.

We then prove that the LDPC-based PRCs of [CG24, AAC+25] satisfy targeted low opening capacity (Lemma 2), and consequently cannot yield UC-secure realizations against adaptive corruption.

Lemma 2 (Informal). *A pseudorandom code based on low-density parity-check (LDPC) codes satisfies targeted low opening capacity (TLOC), and therefore does not admit a decoder-non-committing pseudorandom code.*

Heuristic Constructions for Non-Committing PRC. The positive direction is not vacuous: the key idea is to enlarge the decoder's accepting set so that dummy codewords can be absorbed apart from the true ones, granting the *equivocation* needed to open them after corruption.

In Section 6, we give candidate secret-key and public-key compilers to lift error-correcting codes to admit decoder-non-committing pseudorandom codes, which we prove secure assuming an "ideal" error-correcting code. The secret-key compiler in Section 6.1 uses a *puncturable PRF* [BW13] together with an *indistinguishability obfuscator* (iO) [BGI+01], obtaining equivocality by re-obfuscating the acceptance program after the dummies are drawn; the public-key compiler in Section 6.2 uses a *smooth projective hash function* (SPHF) [CS02] over a hard subset-membership language, whose hashing key retains, given the projection key, enough residual freedom to be opened consistently.

A Dichotomy for Public-Key Fresh-Codeword Explanation. In Section 7 we isolate an issue orthogonal to adaptive corruption: it arises in the public-key setting even with no corrupted party. Since the encoding key is public, a fresh codeword may be generated outside the honest encoding interface yet accepted by an honest decoder, and the ideal-process adversary must explain it to $\mathcal{F}_{\mathsf{PRC}}$ by returning

the same decoded message. This message cannot in general be computed from (ek, c) alone, since any such public extractor would violate pseudorandomness (Lemma 4); hence any realization must provide $\mathcal{S}_{\mathsf{PRC}}$ a trapdoor for ek or rely on an idealized setup (Proposition 7). Consequently, any publicly verifiable verdict contradicts pseudorandomness, making PRC-based watermark detection inherently designated-verifier.

2 Preliminaries on Pseudorandom Code

Since pseudorandom (error-correcting) codes are a recent primitive, we recall their syntax and basic security notions, following [CG24, AAC+25].

Definition 1 (Pseudorandom (Error-Correcting) Code [CG24]). *Let Σ be a message space and $\Sigma^\star$ be the codeword space. A pseudorandom (error-correcting) code* PRC *is a triple of algorithms* $\mathsf{PRC} = (\mathtt{KGen}, \mathtt{Encode}, \mathtt{Decode})$

- $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathtt{KGen}(1^\lambda)$*: on the input of a security parameter λ, output a pair of (public) encoding key* ek *and decoding key* dk[1]*.*
- $c \leftarrow \mathtt{Encode}(\mathsf{ek}, m)$*: on the input of a (public) encoding key* ek *and a string $m \in \Sigma$, output a codeword $c \in \Sigma^\star$.*
- $m/\perp \leftarrow \mathtt{Decode}(\mathsf{dk}, x)$*: on the input of a decoding key* dk *and a (perturbed) codeword x, output a message $m \in \Sigma$ or a symbol $\perp$ indicating decoding failure.*

Definition 2 (Secret-key (Adaptive) δ-Robustness [AAC+25]). *For any $\lambda \in \mathbb{N}$, any message $m \in \Sigma$, a distance function* $\mathsf{dist} : \{0,1\}^* \times \{0,1\}^* \to \Delta$*, and an error bound $\delta \in \Delta$*

$$\Pr\left[\begin{array}{l|l} \exists (m,c) \in \mathsf{tr} : \mathsf{dist}(c,x) < \delta & (\mathsf{ek},\mathsf{dk}) \leftarrow \mathtt{KGen}(1^\lambda), \\ \wedge\, \mathtt{Decode}(\mathsf{dk}, x) \neq m & x \leftarrow \mathcal{A}^{\mathtt{Encode}(\mathsf{ek},\cdot)}(1^\lambda) \end{array}\right] \leq \mathsf{negl}(\lambda),$$

where tr *is the trace that $\mathcal{A}$ adaptively queries to* $\mathtt{Encode}(\mathsf{ek}, \cdot)$*.*

Definition 3 (Public-key (Adaptive) δ-Robustness [AAC+25]). *For any $\lambda \in \mathbb{N}$ and any message $m \in \Sigma$, a distance function* $\mathsf{dist} : \{0,1\}^* \times \{0,1\}^* \to \Delta$*, and an error bound $\delta \in \Delta$,*

$$\Pr\left[\begin{array}{l|l} \mathsf{dist}(x,c) < \delta & (\mathsf{ek},\mathsf{dk}) \leftarrow \mathtt{KGen}(1^\lambda), \\ \wedge\, m \neq m' & (m, x, r) \leftarrow \mathcal{A}(1^\lambda, \mathsf{ek}), \\ & c \leftarrow \mathsf{PRC}.\mathtt{Encode}(\mathsf{ek}, m; r), \\ & m' \leftarrow \mathsf{PRC}.\mathtt{Decode}(\mathsf{dk}, x), \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Definition 4 (Pseudorandomness [CG24]). *For any $\lambda \in \mathbb{N}$ and any polynomial-time adversary $\mathcal{A}$,*

$$\left| \Pr_{(\mathsf{ek},\mathsf{dk}) \leftarrow \mathtt{KGen}(1^\lambda)} \left[\mathcal{A}^{\mathtt{Encode}(\mathsf{ek},\cdot)}(1^\lambda, \mathsf{ek}\ ^{[2]}) = 1 \right] - \Pr_{\mathcal{U}} \left[\mathcal{A}^{\mathcal{U}}(1^\lambda) = 1 \right] \right| \leq \mathsf{negl}(\lambda).$$

[1] For secret-key PRC, we assume $\mathsf{ek} = \mathsf{dk}$.

Definition 5 (Soundness [CG24]). *For any $\lambda \in \mathbb{N}$ and a fixed codeword $c \in \Sigma^\star$ chosen independently of* $(\mathsf{ek}, \mathsf{dk})$*,*

$$\Pr_{(\mathsf{ek},\mathsf{dk}) \leftarrow \mathtt{KGen}(1^\lambda)} [\ \mathtt{Decode}(\mathsf{dk}, c) \neq \perp\ ] \leq \mathsf{negl}(\lambda)$$

3 UC Ideal Functionality of PRC

We recall the UC framework of Canetti [Can01], only as needed. Security compares the *real* execution $\mathbf{REAL}_{\Pi,\mathcal{A}}$, in which parties run a protocol Π with an adversary $\mathcal{A}$, against the *ideal* execution $\mathbf{IDEAL}_{\mathcal{F},\mathcal{S}}$, in which they relay inputs to an *ideal functionality* $\mathcal{F}$ realizing the task by fiat, with an *ideal-process adversary* $\mathcal{S}$ in place of $\mathcal{A}$. Both are driven by an *environment* $\mathcal{Z}$ that chooses inputs, sees outputs, and talks to the adversary; we say Π *realizes* $\mathcal{F}$ if for every $\mathcal{A}$ there exists $\mathcal{S}$ such that $\mathbf{REAL}_{\Pi,\mathcal{A}} \approx \mathbf{IDEAL}_{\mathcal{F},\mathcal{S}}$ for every PPT $\mathcal{Z}$ i.e., $\mathcal{Z}$ cannot distinguish between the two worlds. Corruption is instructed by $\mathcal{Z}$, and a corrupted party surrenders its local state and subsequent behavior to the adversary, either *non-adaptively*, with the corrupted set fixed before the session, or *adaptively*, during the execution. The payoff is the composition theorem, which preserves security when Π is used as a subroutine of a larger protocol, and which the standalone notions of [AAC+25] do not provide.

3.1 Recalling PRC Standalone Security

We recall the standalone ideal-security and CCA-security notions of [AAC+25]. For ease of formalization, we define a function $\mathsf{Near}^{\delta}_{\mathcal{Q}}(\cdot)$ that finds the nearby code in a set $\mathcal{Q}$.

Definition 6 (Nearby Code). *For a set $\mathcal{Q}$ of message-codeword pairs (m, c), we define*

$$\mathsf{Near}^{\delta}_{\mathcal{Q}}(x) \coloneqq \{(m, c) \in \mathcal{Q} : \mathsf{dist}(x, c) < \delta\}.$$

The security games of [AAC+25] do not explicitly handle decoding queries x close to transcript codewords of different messages. In Fig. 1, we make this case explicit, returning $\perp$. This is immaterial for the parameters of [AAC+25], where such ambiguity occurs only with negligible probability. Indeed, otherwise it would violate robustness for one of the encoded messages. Making it explicit aids the UC formalization by separating uniquely explained queries from ambiguous queries (which abort honest decoding) and fresh queries (delegated to the ideal-process adversary).

Definition 7 (Secret-Key Ideal Security [AAC+25]). *A secret-key pseudorandom code* PRC *is* Ideal *secure if for any PPT adversary $\mathcal{A}$,*

$$\mathbf{Adv}^{\mathrm{Ideal}}_{\mathsf{PRC}}(\mathcal{A}) \coloneqq 2 \cdot \left| \Pr[\mathbf{G}^{\mathrm{Ideal}}_{\mathsf{PRC},\mathsf{dist},\delta}(\mathcal{A}) \Rightarrow 1] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

[2] ek is disclosed to $\mathcal{A}$ only for public-key PRC.

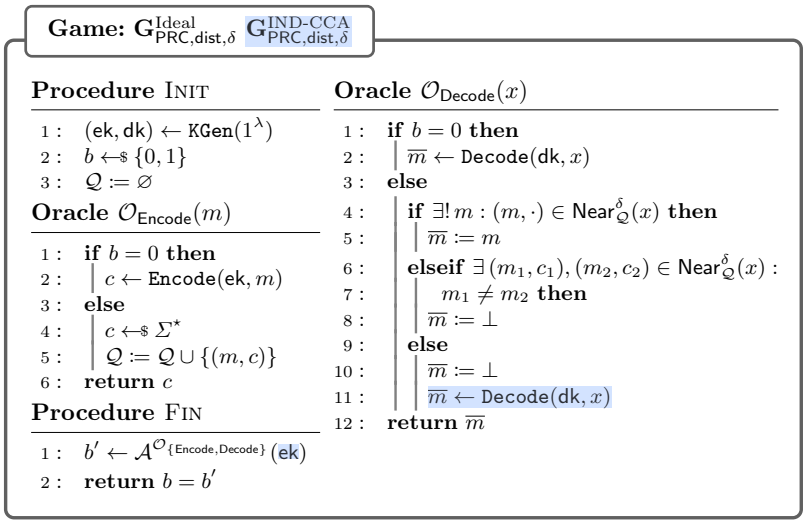

Fig. 1: Ideal-security game for a secret-key pseudorandom code and IND-CCA security game for a public-key pseudorandom code adapted from [AAC+25]. The highlighted part is exclusively for IND-CCA security.

Definition 8 (Public-Key IND-CCA Security [AAC+25]). *A public-key pseudorandom code* PRC *is* IND-CCA *secure if for any PPT adversary* $\mathcal{A}$*,*

$$\mathbf{Adv}^{\text{IND-CCA}}_{\mathsf{PRC}}(\mathcal{A}) := 2 \cdot \left| \Pr[\mathbf{G}^{\text{IND-CCA}}_{\mathsf{PRC},\mathsf{dist},\delta}(\mathcal{A}) \Rightarrow 1] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

3.2 Ideal Functionality $\mathcal{F}_{\mathsf{PRC}}$

We define $\mathcal{F}_{\mathsf{PRC}}$ as the UC analogue of the standalone ideal-security and CCA-style experiments of Fig. 1. As in those games, it maintains an issued table $\mathsf{Issued}[\mathsf{sid}]$ of honestly generated codeword-message pairs and answers a decoding query by comparing it against the δ-neighborhoods of issued codewords: a query close to a unique issued message decodes to that message, while one close to codewords for different messages is ambiguous and aborts.

The main new feature of the UC formalization is that the encoder $\mathcal{P}_{\mathsf{ecd}}$ and decoder $\mathcal{P}_{\mathsf{dcd}}$ are explicit parties whose corruption status affects what $\mathcal{F}_{\mathsf{PRC}}$ must model. In the standalone games the transcript is controlled entirely by the challenger, so every encoding query is recorded and the adversary acts only through oracle calls; this transcript-completeness can fail once parties are corrupted.

Fresh Codewords under Corruption. We first illustrate the effect of corruption through a fresh-codeword query with x, i.e. one with $\mathsf{Near}^\delta_{\mathsf{Issued}[\mathsf{sid}]}(x) = \varnothing$, which

[3] We assume that in the real-world protocol $\mathcal{P}_{\mathsf{dcd}}$ aborts if decoded to $\perp$.

Functionality: $\mathcal{F}_{\mathsf{PRC}}^{\delta,\mathsf{dist}}$

Parameters: an error evaluation function $\mathsf{dist} : \{0,1\}^* \times \{0,1\}^* \to \Delta$; a maximum error $\delta \in \Delta$.
Participants: an encoder $\mathcal{P}_{\mathsf{ecd}}$; a decoder $\mathcal{P}_{\mathsf{dcd}}$; an ideal-process adversary $\mathcal{S}_{\mathsf{PRC}}$.
Initialization: Set Issued as an empty table for all sid; set $\mathsf{Corrupt} := \varnothing$.

– $\langle \texttt{KeyGen} : \mathsf{sid} \rangle$ from $\mathcal{P}_{\mathsf{dcd}}$ or $\mathcal{P}_{\mathsf{ecd}}$:
1. If there is a record $(\mathsf{sid}, \mathsf{ek})$ sid is marked as activated, abort.
2. Otherwise, send $\langle \texttt{KeyGen} : \mathsf{sid} \rangle$ to $\mathcal{S}_{\mathsf{PRC}}$
3. Mark sid as activated when $\mathcal{S}_{\mathsf{PRC}}$ responds.
4. Record $(\mathsf{sid}, \mathsf{ek})$ when $\mathcal{S}_{\mathsf{PRC}}$ returns ek.
5. Output $\{\texttt{"key"} : \mathsf{sid}, \mathsf{ek}\}$ to $\mathcal{P}_{\mathsf{ecd}}$ and $\mathcal{P}_{\mathsf{dcd}}$.

– $\langle \texttt{Encode} : \mathsf{ecd}, \mathsf{sid}, \mathsf{ek}, m \rangle$ from $\mathcal{P}_{\mathsf{ecd}}$
1. If there is no record $(\mathsf{sid}, \mathsf{ek})$ sid is not marked as activated and $\mathsf{ecd} \notin \mathsf{Corrupt}$, abort.
2. If $\mathsf{ecd} \notin \mathsf{Corrupt}$ and $\mathsf{dcd} \notin \mathsf{Corrupt}$:
   a. Send $\langle \texttt{Encode} : \mathsf{ecd}, \mathsf{sid}, \mathsf{ek} \rangle$ to $\mathcal{S}_{\mathsf{PRC}}$.
   b. Sample a random string $c \leftarrow\!\!\$\ \Sigma^\star$ from the codeword space $\Sigma^\star$.
   c. Add (m, c) to $\mathsf{Issued}[\mathsf{sid}]$.
   d. Output $\{\texttt{"code"} : \mathsf{ecd}, \mathsf{sid}, c\}$ to $\mathcal{P}_{\mathsf{ecd}}$.
3. Else: // $\mathsf{ecd} \in \mathsf{Corrupt}$ or $\mathsf{dcd} \in \mathsf{Corrupt}$.
   a. Send $\langle \texttt{Encode} : \mathsf{ecd}, \mathsf{sid}, \mathsf{ek}, m \rangle$ to $\mathcal{S}_{\mathsf{PRC}}$.
   b. Output $\{\texttt{"code"} : \mathsf{ecd}, \mathsf{sid}, c\}$ to $\mathcal{P}_{\mathsf{ecd}}$ when $\mathcal{S}_{\mathsf{PRC}}$ returns c.

– $\langle \texttt{Decode} : \mathsf{dcd}, \mathsf{sid}, x \rangle$ from $\mathcal{P}_{\mathsf{dcd}}$
1. If there is no record $(\mathsf{sid}, \mathsf{ek})$ sid is not marked as activated and $\mathsf{dcd} \notin \mathsf{Corrupt}$, abort.
2. If $\exists! \, m : (m, \cdot) \in \mathsf{Near}^{\delta}_{\mathsf{Issued}[\mathsf{sid}]}(x)$:
   a. If $\mathsf{dcd} \notin \mathsf{Corrupt}$, output $\{\texttt{"word"} : \mathsf{dcd}, \mathsf{sid}, x, m\}$ to $\mathcal{P}_{\mathsf{dcd}}$.
   b. Else: // $\mathsf{dcd} \in \mathsf{Corrupt}$.
      i. Output $\langle \texttt{Decode} : \mathsf{dcd}, \mathsf{sid}, x, m \rangle$ to $\mathcal{S}_{\mathsf{PRC}}$.
      ii. Output $\{\texttt{"word"} : \mathsf{dcd}, \mathsf{sid}, x, m^\star\}$ to $\mathcal{P}_{\mathsf{dcd}}$ when $\mathcal{S}_{\mathsf{PRC}}$ returns $m^\star$.
3. Else if $\exists \, m_1, m_2 : (m_1, \cdot), (m_2, \cdot) \in \mathsf{Near}^{\delta}_{\mathsf{Issued}[\mathsf{sid}]}(x) \wedge m_1 \neq m_2$:
   a. If $\mathsf{dcd} \notin \mathsf{Corrupt}$, abort
   b. Else: // $\mathsf{dcd} \in \mathsf{Corrupt}$.
      i. Output $\left\langle \texttt{Decode} : \mathsf{dcd}, \mathsf{sid}, x, \mathsf{Near}^{\delta}_{\mathsf{Issued}[\mathsf{sid}]}(x) \right\rangle$ to $\mathcal{S}_{\mathsf{PRC}}$.
      ii. Output $\{\texttt{"word"} : \mathsf{dcd}, \mathsf{sid}, x, m^\star\}$ to $\mathcal{P}_{\mathsf{dcd}}$ when $\mathcal{S}_{\mathsf{PRC}}$ returns $m^\star$.
4. Else: // Fresh unseen codeword i.e., $\mathsf{Near}^{\delta}_{\mathsf{Issued}[\mathsf{sid}]}(x) = \varnothing$.
   a. If $\mathsf{dcd} \notin \mathsf{Corrupt} \wedge \mathsf{ecd} \notin \mathsf{Corrupt}$, abort.
   b. Output $\langle \texttt{Decode} : \mathsf{dcd}, \mathsf{sid}, x \rangle$ to $\mathcal{S}_{\mathsf{PRC}}$.
   c. When $\mathcal{S}_{\mathsf{PRC}}$ returns m, abort if $m = \perp \wedge \mathsf{dcd} \notin \mathsf{Corrupt}$ [3].
   d. Output $\{\texttt{"word"} : \mathsf{dcd}, \mathsf{sid}, x, m\}$ to $\mathcal{P}_{\mathsf{dcd}}$.

– $\langle \texttt{Corrupt} : \mathsf{pid}, \mathsf{sid} \rangle$ from $\mathcal{S}_{\mathsf{PRC}}$: // $\mathsf{pid} \in \{\mathsf{ecd}, \mathsf{dcd}\}$
1. Label $\mathcal{P}_{\mathsf{pid}}$ as corrupted by setting $\mathsf{Corrupt} := \mathsf{Corrupt} \cup \{\mathsf{pid}\}$.
2. If $\mathsf{pid} = \mathsf{dcd}$ or $\mathsf{pid} = \mathsf{ecd}$, disclose all pairs in $\mathsf{Issued}[\mathsf{sid}]$ to $\mathcal{S}_{\mathsf{PRC}}$.

Fig. 2: An ideal functionality $\mathcal{F}_{\mathsf{PRC}}$ for pseudorandom codes lifted from oracle-based ideal security and CCA security in [AAC+25]. The functionality is parameterized by an error evaluation function $\mathsf{dist} : \{0,1\}^* \times \{0,1\}^* \to \Delta$ that evaluates the error between two codewords, and a maximum error bound $\delta \in \Delta$. The highlighted part appears exclusively when PRC is public-key i.e., ek is public. The framed part appears exclusively when PRC is secret-key.

is where UC security may diverge from the standalone notion. In the secret-key setting, the single key $\mathsf{ek} = \mathsf{dk}$ is secret, so only a holder of the key can produce valid codewords. While both parties remain uncorrupted, a fresh valid codeword outside $\mathsf{Issued}[\mathsf{sid}]$ should therefore be infeasible to generate, so $\mathcal{F}_{\mathsf{PRC}}$ aborts, mirroring the return of $\perp$ in $\mathcal{O}_{\mathsf{Decode}}$ in Fig. 1. Once $\mathcal{P}_{\mathsf{ecd}}$ is corrupted, however, the key leaks and the interface $\langle \mathtt{Encode} : \cdot \rangle$ can be bypassed by encoding locally with ek, producing valid codewords outside $\mathsf{Issued}[\mathsf{sid}]$ that $\mathcal{S}_{\mathsf{PRC}}$ must then explain. Corruption thus plays in the secret-key setting the role that publicity of ek plays in the public-key setting, which makes fresh valid codewords possible outside $\mathcal{F}_{\mathsf{PRC}}$. Here UC security diverges from the standalone ideal-security game, whose decoding oracle rejects every word outside the transcript because the adversary never holds the key. Once the key leaks, $\mathcal{F}_{\mathsf{PRC}}$ instead delegates such a codeword to $\mathcal{S}_{\mathsf{PRC}}$, which holds the same key and hence can answer it with the real decoder.

The public-key setting is more permissive still. Since ek is public, such codewords may arise even without corruption, so the public-key functionality delegates *every* fresh decoding query to $\mathcal{S}_{\mathsf{PRC}}$, mirroring the fallback-to-real-decoder behavior of the standalone CCA experiment in Fig. 1. As in that experiment, the answer to a fresh query is not added to $\mathsf{Issued}[\mathsf{sid}]$. A fresh word that decodes successfully is not necessarily an honest codeword, and may be an already perturbed one. Hence, recording it would extend the accepting region and accept even out-of-boundary words that the real decoder rejects. This differs from the public-key encryption functionality of [CDG+18], which records the answer to a fresh decryption query but reuses it only for the same ciphertext, which is harmless memoization since decryption is deterministic.

Decoder-State Exposure. We consider $\mathcal{F}_{\mathsf{PRC}}$ in a *decoder-state exposure* model, which becomes critical under *adaptive* corruption: corrupting $\mathcal{P}_{\mathsf{dcd}}$ yields a decoder state (e.g. dk) with which the adversary can locally evaluate $\mathtt{Decode}(\mathsf{dk}, \cdot)$ on issued codewords. Hence, when $\mathcal{F}_{\mathsf{PRC}}$ discloses $\mathsf{Issued}[\mathsf{sid}]$ to $\mathcal{S}_{\mathsf{PRC}}$ after corruption, $\mathcal{S}_{\mathsf{PRC}}$ must supply a decoder state consistent with the dummy codewords sampled while the parties were honest, which is the source of the decoder-opening obstruction of Section 5. This is the faithful model since $\mathsf{PRC}.\mathtt{Decode}(\cdot)$ is a deterministic function of dk, so a corrupted decoder that retains its long-term key can evaluate it at will. Once $\mathcal{P}_{\mathsf{dcd}}$ is corrupted, codewords are no longer hidden from the adversary, who can decode them with the exposed state, so $\mathcal{F}_{\mathsf{PRC}}$ stops sampling dummy codewords and lets $\mathcal{S}_{\mathsf{PRC}}$ produce subsequent ones, as for a corrupted encoder; only the dummy codewords issued before corruption must be explained.

We also assume *erasure* of past encoder randomness and transient *encoder* state. This is mild, since the randomness of an encoding is used only to produce its codeword and is never needed later, e.g. for decoding. Corrupting $\mathcal{P}_{\mathsf{ecd}}$ then reveals only the current long-term encoding state, and no erased randomness needs to be explained. In the secret-key case, this state is the shared key $\mathsf{ek} = \mathsf{dk}$, so $\mathcal{S}_{\mathsf{PRC}}$ answers the corruption with the same opened decoder state as for a corruption of $\mathcal{P}_{\mathsf{dcd}}$. Since $\mathcal{P}_{\mathsf{dcd}}$ may still be honest, this opened state must

moreover decode the δ-neighborhood of each dummy codeword to the message that $\mathcal{F}_{\mathsf{PRC}}$ returns from $\mathsf{Issued}[\mathsf{sid}]$. If $\mathsf{ek} \neq \mathsf{dk}$ are both secret, e.g. in the multi-bit transform of [AAC+25], exposing ek under erasure only requires codewords to remain pseudorandom given ek, as for public-key PRCs in Definition 4. Without erasure, the revealed randomness would let the adversary re-encode each dummy and compare, so a realization would need a full non-committing notion that additionally opens the encoding randomness, i.e., for each dummy codeword c_i of message m_i, randomness $\rho_i^\star$ with $c_i = \mathsf{PRC.Encode}(\mathsf{ek}, m_i\,;\,\rho_i^\star)$. Either way, such requirements only strengthen what a realization must achieve, so our impossibility results in Section 5.2, which rely on decoder corruption alone, apply unchanged.

Table 1: Additional requirements imposed by corrupting $\mathcal{P}_{\mathsf{ecd}}$, by key structure and erasure assumption.

Key Structure	With Erasure	Without Erasure
Secret-key, $\mathsf{ek} = \mathsf{dk}$ (this work)	IND-OPEN + δ-opening robustness	+ open the encoding randomness
Secret-key, $\mathsf{ek} \neq \mathsf{dk}$	Pseudorandomness given ek	
Public-key	None (ek is public)	

Table 1 summarizes the requirements imposed by encoder corruption. Without erasure, every row additionally requires opening the encoding randomness, independently of the key structure. Under erasure, a *non-decoding* ek only requires pseudorandomness given ek, which public-key PRCs satisfy by definition. For $\mathsf{ek} = \mathsf{dk}$, this is impossible, since $\mathtt{Decode}(\mathsf{ek}, \cdot)$ accepts codewords but, by soundness, rejects uniform strings. This is thus the only case that calls for new security notions, which we introduce in Section 5.1: *decoder-opening indistinguishability* (IND-OPEN) and δ-opening robustness. The former replaces pseudorandomness given ek by letting the key be opened after the dummy codewords are drawn, and reduces to it when $\mathsf{ek} \neq \mathsf{dk}$.

We stress that our impossibility results are specific to this model. A decoder that erases dk after each activation, or that evaluates $\mathtt{Decode}$ only through a stateful token, never hands $\mathcal{Z}$ a locally evaluable state, so no opening is demanded and the obstruction of Section 5 does not arise. Such models evade the barrier only by strengthening the erasure or setup assumption, and they no longer capture a decoder holding a long-term decoding key, which is the setting of interest for watermark detection.

4 Equivalence under Non-Adaptive Corruption

We relate the standalone oracle notions of [AAC+25] to $\mathcal{F}_{\mathsf{PRC}}$. The only new feature is the explicit corruption status of $\mathcal{P}_{\mathsf{ecd}}$ and $\mathcal{P}_{\mathsf{dcd}}$, which we fix *non-adaptively*: $\mathsf{Corrupt} \subseteq \{\mathsf{ecd}, \mathsf{dcd}\}$ is determined before the session, so each party is adversarial throughout or honest throughout. Since corruption is fixed before any codeword is sampled, $\mathcal{S}_{\mathsf{PRC}}$ never has to explain an ideal codeword $c \leftarrow\!\!\$\ \Sigma^\star$

under local state revealed only later. Under this restriction $\mathcal{F}_{\mathsf{PRC}}$ is the UC analogue of the corresponding standalone experiment: honest encoding queries are answered by sampling fresh $c \leftarrow\!\!\$\ \Sigma^\star$ and recording (m, c) in $\mathsf{Issued}[\mathsf{sid}]$, honest decoding by the table-first rule, and a party corrupted from the outset has its behavior generated by the adversary in both worlds, with $\mathcal{F}_{\mathsf{PRC}}$ merely exposing the transcript needed for simulation. Thus the secret-key setting matches the ideal-security game (with $\mathcal{Q} \leftrightarrow \mathsf{Issued}[\mathsf{sid}]$), and the public-key setting the random branch ($b = 1$) of IND-CCA, up to the standard oracle/UC translation, analogous to [CHK05, CLNS17].

Theorem 1 (Equivalence under Non-Adaptive Corruption, Secret-Key). *Let* $\mathsf{PRC} = (\mathtt{KGen}, \mathtt{Encode}, \mathtt{Decode})$ *be a secret-key pseudorandom code, and let* Π_{prc} *be the natural protocol in which* $\mathcal{P}_{\mathsf{ecd}}$ *and* $\mathcal{P}_{\mathsf{dcd}}$ *run* $\mathsf{PRC}.\mathtt{Encode}(\mathsf{ek}, \cdot)$ *and* $\mathsf{PRC}.\mathtt{Decode}(\mathsf{dk}, \cdot)$ *under honestly generated keys* $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PRC}.\mathtt{KGen}(1^\lambda)$. *Then* Π_{prc} *realizes* $\mathcal{F}_{\mathsf{PRC}}$ *against non-adaptive corruption if and only if* PRC *satisfies secret-key ideal security (Fig. 1), up to the standard translation between UC environments and oracle-game adversaries.*

Proof (Sketch). Since $\mathsf{Corrupt}$ is fixed before the session, we distinguish three cases. If no party is corrupted, the equivalence is the standard oracle/UC correspondence: a distinguishing environment $\mathcal{Z}$ yields a standalone adversary $\mathcal{A}_{sa}$ against the ideal security of PRC that answers $\langle \mathtt{Encode} : \cdot \rangle$ and $\langle \mathtt{Decode} : \cdot \rangle$ via $\mathcal{O}_{\mathsf{Encode}}$ and $\mathcal{O}_{\mathsf{Decode}}$; the branches $b = 0, 1$ reproduce **REAL** and **IDEAL**, so $\mathcal{A}_{sa}$ inherits $\mathcal{Z}$'s advantage, and the converse is symmetric.

If $\mathcal{P}_{\mathsf{dcd}}$ is corrupted, whether or not $\mathcal{P}_{\mathsf{ecd}}$ is, $\mathcal{F}_{\mathsf{PRC}}$ lets $\mathcal{S}_{\mathsf{PRC}}$ produce every codeword and the adversary performs every decoding, so $\mathcal{S}_{\mathsf{PRC}}$ simply runs the protocol with the key it generated and the two executions coincide. If only $\mathcal{P}_{\mathsf{ecd}}$ is corrupted, $\mathcal{Z}$ holds $\mathsf{ek} = \mathsf{dk}$ and every codeword is produced outside $\langle \mathtt{Encode} : \cdot \rangle$: in **REAL**, $\mathcal{P}_{\mathsf{dcd}}$ outputs $\mathtt{Decode}(\mathsf{dk}, c)$, while in **IDEAL**, $\mathcal{F}_{\mathsf{PRC}}$ delegates the fresh c to $\mathcal{S}_{\mathsf{PRC}}$, which holds the same key and returns the same value. This is faithful because non-adaptive corruption exposes the key before any codeword is sampled, so no ideal codeword is committed before $\mathcal{S}_{\mathsf{PRC}}$ can explain it. Hence corruption adds no assumption, and ideal security is used only when no party is corrupted. □

We defer the full proof of Theorem 1 to Appendix C.1.

Theorem 2 (Equivalence under Non-Adaptive Corruption, Public-Key). *Let* $\mathsf{PRC} = (\mathtt{KGen}, \mathtt{Encode}, \mathtt{Decode})$ *be a public-key pseudorandom code, and let* Π_{prc} *be the natural protocol in which* $\mathcal{P}_{\mathsf{ecd}}$ *and* $\mathcal{P}_{\mathsf{dcd}}$ *run* $\mathsf{PRC}.\mathtt{Encode}(\mathsf{ek}, \cdot)$ *and* $\mathsf{PRC}.\mathtt{Decode}(\mathsf{dk}, \cdot)$ *under honestly generated keys* $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PRC}.\mathtt{KGen}(1^\lambda)$, *with* ek *public. Then* Π_{prc} *realizes* $\mathcal{F}_{\mathsf{PRC}}$ *against non-adaptive corruption if and only if* PRC *satisfies* IND-CCA *security (Fig. 1), up to the standard translation between UC environments and oracle-game adversaries.*

Proof (Sketch). The proof largely follows Theorem 1, the only delicate point being fresh decoding queries: since ek is public and adversarially generated

codewords may decode successfully, such queries cannot simply be rejected. Accordingly, $\mathcal{S}_{\mathsf{PRC}}$ samples $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathtt{KGen}(1^\lambda)$, returns only ek to $\mathcal{F}_{\mathsf{PRC}}$, and keeps dk as a non-public trapdoor, answering each delegated fresh query x by $m \leftarrow \mathsf{PRC}.\mathtt{Decode}(\mathsf{dk}, x)$, which is exactly the fallback-to-real-decoder behavior of the random branch of the IND-CCA game. Non-adaptive corruption fixes the decoder's status in advance, so dk is never demanded as corrupted local state and the trapdoor stays hidden throughout. □

We defer the full proof of Theorem 2 to Appendix C.2.

5 Separation under Adaptive Corruption

The equivalence above breaks under *adaptive corruption* (with decoder-state exposure). Suppose an uncorrupted $\mathcal{P}_{\mathsf{ecd}}$ is activated with $\langle \mathtt{Encode} : \cdot \rangle$ on a message m. In the ideal execution, $\mathcal{F}_{\mathsf{PRC}}$ samples a dummy codeword $c \leftarrow\!\!\$\ \Sigma^\star$ and records (m, c) in $\mathsf{Issued}[\mathsf{sid}]$. If $\mathcal{P}_{\mathsf{dcd}}$ is later corrupted, the environment $\mathcal{Z}$ obtains dk and can test whether $\mathsf{PRC}.\mathtt{Decode}(\mathsf{dk}, c) = m$. In the real execution this succeeds by robustness, since $c = \mathsf{PRC}.\mathtt{Encode}(\mathsf{ek}, m)$; in the ideal execution c was sampled independently of dk, so by soundness it decodes to $\perp$ except with negligible probability.

This is the PRC analogue of the *non-committing encryption* [CFGN96] obstruction for UC-secure PKE under adaptive corruption [CHK05, CLNS17], where the simulator must produce dummy ciphertexts that can later be *opened* consistently as encryptions of chosen messages. The obstruction affects both public-key PRC with IND-CCA security and secret-key PRC with ideal security, since in both, honest codewords are dummies before corruption.

5.1 Non-Committing Pseudorandom Code

For PRCs, we call the corresponding object a *decoder-non-committing pseudorandom code*. The distinction is that a ciphertext opening is pointwise, i.e., opens to an exact ciphertext, whereas a PRC opening must explain codewords. In the secret-key case with $\mathsf{ek} = \mathsf{dk}$, where corrupting the encoder leaks dk while the decoder remains honest, it must moreover be robust, that is, after opening a dummy codeword c to a message m, not only c itself but also the entire δ-neighborhood of c must decode consistently to m, except in the ambiguous-overlap cases already handled by $\mathcal{F}_{\mathsf{PRC}}$. In Definition 9, we first define the syntax of a decoder-non-committing pseudorandom code.

We stress that an NC-PRC is not only a proof artifact, since $\mathsf{PRC} = (\mathtt{KGen}, \mathtt{Encode}, \mathtt{Decode})$ itself must first be designed properly so that it is efficient and admits a working opening algorithm simultaneously. In addition, all algorithms of NC-PRC must be PPT so that $\mathcal{S}_{\mathsf{PRC}}$ can run them in the ideal world. An opening that merely exists but cannot be obtained efficiently, e.g., finding a consistent decoder state by exhaustive search, does not yield UC security.

Definition 9 (Decoder-Non-Committing Code). *Let* $\mathsf{PRC} = (\mathtt{KGen}, \mathtt{Encode}, \mathtt{Decode})$ *be a pseudorandom code. We say that* PRC *admits a* decoder-non-committing pseudorandom code *if there exist PPT algorithms* $\mathsf{NC\text{-}PRC} = (\mathtt{SimKGen}, \mathtt{SimCode}, \mathtt{OpenDec})$ *such that*

- $(\mathsf{pp}, \mathsf{st}, \tau_k) \leftarrow \mathtt{SimKGen}(1^\lambda)$*: on input the security parameter* λ*, outputs public information* pp*, an internal simulation state* st*, and a key-opening trapdoor* τ_k*.*
- $(c, \tau_c) \leftarrow \mathtt{SimCode}(\mathsf{pp})$*: on input the public parameter* pp*, outputs a dummy codeword* c *together with a codeword-opening trapdoor* τ_c*.*
- $\mathsf{dk}^\star \leftarrow \mathtt{OpenDec}(\mathsf{st}, \tau_k, \mathcal{T})$*: on input the simulation state* st*, the key-opening trapdoor* τ_k*, and any opening transcript* $\mathcal{T} = \{(m_i, c_i, \tau_{c_i})\}_{i=1}^{q}$ *for some* $q \in \mathbb{N}$*, outputs a decoder state* $\mathsf{dk}^\star$*.*

We say that NC-PRC *is* compatible with $\mathcal{F}_{\mathsf{PRC}}$ *if* $\mathsf{NC\text{-}PRC}.\mathtt{SimCode}(\mathsf{pp})$ *samples* $c \leftarrow\!\!\$\ \Sigma^\star$ *uniformly and independently of all other values, and* τ_c *only records the index of the call.*

For generality, Definition 9 uses a common syntax for both secret-key and public-key PRCs: in the secret-key case $\mathsf{pp} := \bot$, while in the public-key case pp contains the public encoding key, i.e., $\mathsf{pp} := \mathsf{ek}$. The state st lets $\mathsf{NC\text{-}PRC}.\mathtt{SimCode}(\cdot)$ resume the same execution and produce further dummy codewords. The value τ_c records which simulated $\langle \mathtt{Encode} : \cdot \rangle$ activation, on message m, produced the dummy word c. It carries no cryptographic content and serves only to let $\mathsf{NC\text{-}PRC}.\mathtt{OpenDec}(\cdot)$ match each codeword in the opening transcript to the activation that produced it, for example by recording the index of that activation call.

The reason such an NC-PRC mechanism must exist for adaptive UC security is that: upon adaptive corruption of $\mathcal{P}_{\mathsf{dcd}}$, the ideal-process adversary $\mathcal{S}_{\mathsf{PRC}}$ must be able to produce a decoder state $\mathsf{dk}^\star$ that is disclosed to the environment $\mathcal{Z}$ to explain the dummy codewords issued before corruption. In NC-PRC, this state is produced as $\mathsf{dk}^\star \leftarrow \mathsf{NC\text{-}PRC}.\mathtt{OpenDec}(\mathsf{st}, \tau_k, \mathcal{T})$ for $\mathcal{T} = \{(m_i, c_i, \tau_{c_i})\}_i$ where τ_k is the analogue of τ_c for the simulated key-generation phase and it identifies the internal ideal execution after the simulated $\langle \mathtt{KeyGen} : \cdot \rangle$ activation.

For simplicity, we extend the nearby code of Definition 6 to a *simulated* nearby code that additionally includes the trapdoor, in Definition 10.

Definition 10 (Simulated Nearby Code). *For a simulated opening transcript* $\mathcal{T}$ *of message-codeword-trapdoor tuples* (m, c, τ_c)*, we define the simulated nearby-code set around* x *by*

$$\mathsf{SimNear}^{\delta}_{\mathcal{T}}(x) := \{(m, c, \tau) \in \mathcal{T} : \mathsf{dist}(x, c) < \delta\}.$$

Decoder-Opening Indistinguishability. Under adaptive decoder corruption, the environment $\mathcal{Z}$ observes the codewords produced before corruption and then receives the decoder state revealed after corruption. For both secret-key and public-key PRCs, we therefore require an *indistinguishability* condition, that is, the joint view consisting of dummy codewords and the opened decoder state $\mathsf{dk}^\star$ should be indistinguishable from the view consisting of honestly generated PRC codewords and the real decoder state.

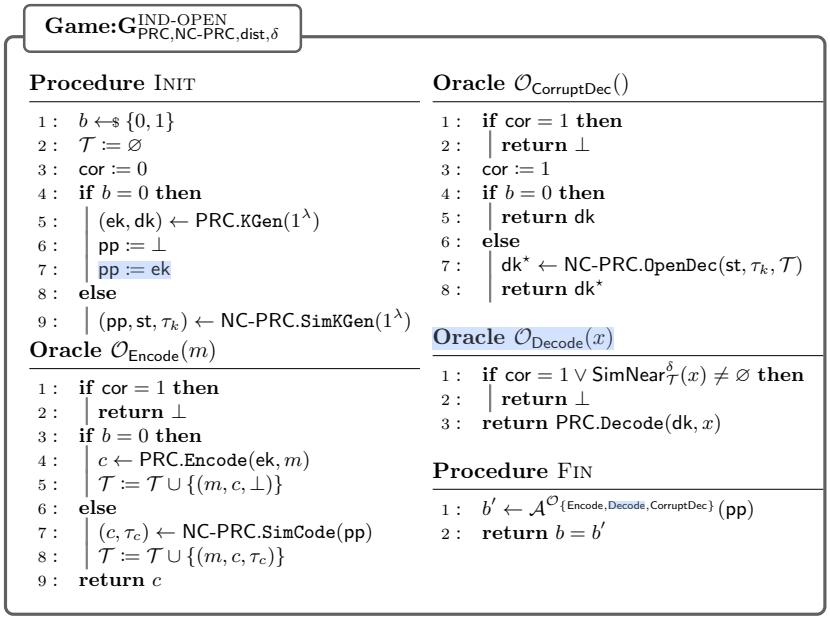

Fig. 3: Decoder-opening indistinguishability (IND-OPEN) game for decoder-non-committing pseudorandom codes. The public value pp is given to the adversary. In public-key PRC, $\mathsf{pp} := \mathsf{ek}$, and in secret-key PRC, $\mathsf{pp} := \bot$. The highlighted parts are exclusively for public-key PRC: the decoding oracle answers fresh words with the real decoder, where in the simulated branch dk is the decoding key contained in st.

Definition 11 (Decoder-Opening Indistinguishability). *Let* $\mathsf{NC\text{-}PRC} = (\mathtt{SimKGen}, \mathtt{SimCode}, \mathtt{OpenDec})$ *be a decoder-non-committing code for a pseudorandom code* $\mathsf{PRC} = (\mathtt{KGen}, \mathtt{Encode}, \mathtt{Decode})$. *We say that* $\mathsf{NC\text{-}PRC}$ *satisfies* decoder-opening indistinguishability (IND-OPEN) *if, for every PPT adversary* $\mathcal{A}$*,*

$$\mathbf{Adv}^{\text{IND-OPEN}}_{\mathsf{PRC},\mathsf{NC\text{-}PRC}}(\mathcal{A}) := 2 \cdot \left| \Pr\left[\mathbf{G}^{\text{IND-OPEN}}_{\mathsf{PRC},\mathsf{NC\text{-}PRC},\mathsf{dist},\delta}(\mathcal{A}) \Rightarrow 1\right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

The oracles $\mathcal{O}_{\mathsf{Encode}}$ and $\mathcal{O}_{\mathsf{Decode}}$ return $\bot$ after corruption without loss of generality: from then on, $\mathcal{F}_{\mathsf{PRC}}$ lets $\mathcal{S}_{\mathsf{PRC}}$ produce codewords, and the adversary can compute them itself, from the revealed key in the secret-key case with $\mathsf{ek} = \mathsf{dk}$ and from $\mathsf{pp} = \mathsf{ek}$ in the public-key case, and it can decode with the revealed key.

In the public-key case, the adversary may also decode *fresh* words before corruption, i.e., words not close to any transcript codeword, such as honest encodings of its own under $\mathsf{pp} = \mathsf{ek}$. Both branches answer them with the real decoder: in the simulated branch, the simulation state st output by $\mathsf{NC\text{-}PRC}.\mathtt{SimKGen}(\cdot)$ contains the decoding key dk of the key pair with $\mathsf{pp} = \mathsf{ek}$, as in the random branch of IND-CCA security, and only the opened state $\mathsf{dk}^\star$ is simulated. Words close to transcript codewords are left to IND-CCA security. After corruption, the adversary can re-decode its earlier queries with the opened state, so the game

forces $\mathsf{dk}^\star$ to agree with dk on them, which neither IND-CCA security, which never reveals a key, nor a decoding-free game checks. In the secret-key case, no such oracle is needed, since $\mathcal{F}_{\mathsf{PRC}}$ rejects fresh words while both parties are honest.

Opening Robustness in Secret-Key NC-PRCs. IND-OPEN asks only that the opened view be indistinguishable from the real one. It does not compare the opened state with a decoder that is still honest after the opening. Such a decoder arises in the secret-key case: since $\mathsf{ek} = \mathsf{dk}$, corrupting $\mathcal{P}_{\mathsf{ecd}}$ already reveals the decoder state, while $\mathcal{P}_{\mathsf{dcd}}$ stays honest. The environment $\mathcal{Z}$ then holds the opened state and can decode not only the codeword c output by $\mathcal{F}_{\mathsf{PRC}}$ but also any perturbed word x with $\mathsf{dist}(x, c) < \delta$, which the still honest decoder answers with m from $\mathsf{Issued}[\mathsf{sid}]$, so the two must agree. We capture this consistency requirement by *opening robustness* in Definition 12: once the simulator opens a decoder state, the dummy codewords must decode consistently throughout their δ-neighborhoods.

Definition 12 (δ-Opening (Adaptive) Robustness). *Let* $\mathsf{NC\text{-}PRC} = (\mathtt{SimKGen}, \mathtt{SimCode}, \mathtt{OpenDec})$ *be a decoder-non-committing pseudorandom code for* PRC*. We say that* NC-PRC *satisfies* δ-opening robustness *if, for every PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, the probability*

$$\Pr\left[\begin{array}{l|l} & (\mathsf{pp}, \mathsf{st}, \tau_k) \leftarrow \mathsf{NC\text{-}PRC}.\mathtt{SimKGen}(1^\lambda), \\ \exists!\,(m, \cdot, \cdot) \in \mathsf{SimNear}^{\delta}_{\mathcal{T}}(x) & \mathsf{st}_{\mathcal{A}} \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{Encode}}(\cdot)}(1^\lambda, \mathsf{pp}), \\ \wedge\ \mathsf{PRC}.\mathtt{Decode}(\mathsf{dk}^\star, x) \neq m & \mathsf{dk}^\star \leftarrow \mathsf{NC\text{-}PRC}.\mathtt{OpenDec}(\mathsf{st}, \tau_k, \mathcal{T}), \\ & x \leftarrow \mathcal{A}_2(\mathsf{st}_{\mathcal{A}}, \mathsf{dk}^\star) \end{array}\right]$$

is negligible in λ*, where the oracle* $\mathcal{O}_{\mathsf{Encode}}$ *invokes* $(c, \tau_c) \leftarrow \mathsf{NC\text{-}PRC}.\mathtt{SimCode}(\mathsf{pp})$ *on each* m *adaptively queried by* $\mathcal{A}$*, responds* $\mathcal{A}$ *with* c*, and sets* $\mathcal{T} := \mathcal{T} \cup \{(m, c, \tau_c)\}$ *with* $\mathcal{T}$ *initialized as* $\mathcal{T} := \varnothing$*.*

Opening Robustness in Public-Key NC-PRCs. In the public-key case, only the decoder state is opened, as in receiver-non-committing encryption [JL00, CHK05], and Theorem 3 does not require δ-opening robustness. Before corruption, words close to issued codewords are answered from the table, with which the real decoder agrees by IND-CCA security, and fresh words are answered by the real decoder, as in the decoding oracle of IND-OPEN. The decoder state is exposed only by corrupting $\mathcal{P}_{\mathsf{dcd}}$, and from then on every word, close or fresh, is decoded by running $\mathsf{PRC}.\mathtt{Decode}(\cdot)$ with the revealed state, i.e., the real dk or the opened $\mathsf{dk}^\star$, while no honest decoder answers from the table. Hence IND-OPEN, which makes the revealed state indistinguishable from the real one, suffices. In particular, for a word x close to a dummy codeword of message m_i and fixed before corruption, IND-CCA security gives $\mathsf{PRC}.\mathtt{Decode}(\mathsf{dk}, x) = m_i$ and IND-OPEN carries this over to $\mathsf{PRC}.\mathtt{Decode}(\mathsf{dk}^\star, x) = m_i$, whereas a search with the revealed state for a wrongly decoded word succeeds with the same probability under dk and $\mathsf{dk}^\star$.

Theorem 3 (Characterization of Adaptive UC Security). *Let* $\mathsf{PRC} = (\mathtt{KGen}, \mathtt{Encode}, \mathtt{Decode})$ *be a pseudorandom code, and let* Π_{prc} *be its natural protocol as in Theorems 1 and 2. Then* Π_{prc} *realizes* $\mathcal{F}_{\mathsf{PRC}}$ *against adaptive corruptions with decoder-state exposure*[4] *if and only if* PRC *satisfies ideal security (secret-key) or* IND-CCA *security (public-key) and admits a decoder-non-committing pseudorandom code*[5] *compatible with* $\mathcal{F}_{\mathsf{PRC}}$ *that satisfies* IND-OPEN *and, in the secret-key case,* δ*-opening robustness.*

Proof (Sketch). ($\Rightarrow$) An adaptive realization is in particular a non-adaptive one, so Theorems 1 and 2 give ideal security or IND-CCA security. We extract NC-PRC from $\mathcal{S}_{\mathsf{PRC}}$: NC-PRC.$\mathtt{SimKGen}(\cdot)$ starts an internal ideal execution, NC-PRC.$\mathtt{SimCode}(\cdot)$ samples $c \leftarrow\!\!\$\ \Sigma^\star$ as $\mathcal{F}_{\mathsf{PRC}}$ does, so NC-PRC is compatible, and NC-PRC.$\mathtt{OpenDec}(\cdot)$ corrupts the party holding the decoder state and returns the state fabricated by $\mathcal{S}_{\mathsf{PRC}}$. Running an IND-OPEN adversary as a UC environment gives IND-OPEN, since in the public-key case $\mathcal{S}_{\mathsf{PRC}}$ decodes fresh words with its real key, as the simulated branch does. In the secret-key case, an environment that corrupts only $\mathcal{P}_{\mathsf{ecd}}$, whose key $\mathsf{ek} = \mathsf{dk}$ then leaks, compares its own decoding under $\mathsf{dk}^\star$ with the answers of the still honest decoder, which $\mathcal{F}_{\mathsf{PRC}}$ gives from the table; this yields δ-opening robustness.

($\Leftarrow$) $\mathcal{S}_{\mathsf{PRC}}$ runs NC-PRC.$\mathtt{SimKGen}(\cdot)$, decodes delegated fresh words with the real key in st, and reveals $\mathsf{dk}^\star \leftarrow$ NC-PRC.$\mathtt{OpenDec}(\mathsf{st}, \tau_k, \mathcal{T})$ at the first corruption that exposes the decoder state. By ideal security or IND-CCA security, the honest decoder may answer words close to issued codewords from the table before that corruption. The resulting execution is emulated by an IND-OPEN adversary, exactly in the simulated branch, and in the real branch up to a table answer that differs from the real decoder after the opening. Such an answer needs an honest decoder after the opening, i.e., the secret-key case, where $\mathsf{ek} = \mathsf{dk}$, with only $\mathcal{P}_{\mathsf{ecd}}$ corrupted, and is ruled out by δ-opening robustness and since uniform dummy codewords are rarely within 2δ of each other. □

We defer the full proof of Theorem 3 to Appendix C.3.

In the public-key case, the condition that the ideal-process adversary generates the key pair with PRC.$\mathtt{KGen}(\cdot)$ and decodes fresh words with the real decoder only matters for decoding queries. Without it, the proof of ($\Rightarrow$) still yields NC-PRC.$\{\mathtt{SimKGen}, \mathtt{SimCode}, \mathtt{OpenDec}\}$ satisfying IND-OPEN against adversaries making no decoding queries, which suffices for the impossibility for codes with targeted low opening capacity (Lemma 1), since the adversary in its proof makes no decoding queries.

In the secret-key case, the proof of ($\Rightarrow$) opens the decoder state by corrupting $\mathcal{P}_{\mathsf{ecd}}$, which is needed only to derive δ-opening robustness, since that argument

[4] In the public-key case, the ideal-process adversary generates the key pair with PRC.$\mathtt{KGen}(\cdot)$ and decodes fresh words with the real decoder.

[5] We assume that dist is a metric under which two independent uniform words of $\Sigma^\star$ are within distance 2δ only with negligible probability, so that, as for honest codewords (Fig. 1), queries close to dummy codewords of different messages occur only with negligible probability.

compares the opened state with a decoder that stays honest. By Lemma 1, a PRC with targeted low opening capacity does not admit a compatible decoder-non-committing pseudorandom code that satisfies δ-opening robustness or IND-OPEN, so the impossibility needs only IND-OPEN, which also follows when $\mathcal{P}_{\mathsf{dcd}}$ is corrupted instead, as this exposes $\mathsf{ek} = \mathsf{dk}$ as well. Hence this impossibility, and with it the one for LDPC-based PRCs, relies on decoder corruption alone, as stated in Section 3.

Throughout, the secret-key case is the one with $\mathsf{ek} = \mathsf{dk}$, as in Definition 1. If $\mathsf{ek} \neq \mathsf{dk}$ are both secret, e.g., in the multi-bit transform of [AAC+25], corrupting $\mathcal{P}_{\mathsf{ecd}}$ reveals only ek, which cannot decode. This case is not covered by Theorem 3 and behaves like the public-key case: the dummy codewords must remain pseudorandom given ek (Table 1), fresh valid words appear once ek leaks and must be decoded with the real key consistently with the later opening, and δ-opening robustness is not forced, since dk leaks only when $\mathcal{P}_{\mathsf{dcd}}$ is corrupted.

5.2 Impossibility with Targeted Low Opening Capacity

We next identify a structural condition of the code under which a PRC *cannot* admit a decoder-non-committing pseudorandom code. In Definition 13, we define *targeted low opening capacity* (TLOC). This notion implies that the decoder family has too little capacity to explain random strings, so that the PRC does not admit a decoder-non-committing pseudorandom code and thus cannot be UC-secure under adaptive corruption.

Definition 13 (Targeted Low Opening Capacity). *Let* $\mathsf{PRC} = (\mathtt{KGen}, \mathtt{Encode}, \mathtt{Decode})$ *be a pseudorandom code with message space* Σ *and codeword space* $\Sigma^\star$*. For a message sequence* $\mathbf{m} = (m_1, \ldots, m_q) \in \Sigma^q$*, we say that* PRC *has* $(\mathbf{m}, q)$*-targeted low opening capacity (*TLOC*) if*

$$\Pr_{c_1,\ldots,c_q \leftarrow\$\, \Sigma^\star}\left[\exists\, \mathsf{dk}^\star : \forall i \in [q],\ \mathtt{Decode}(\mathsf{dk}^\star, c_i) = m_i\right] \leq \mathsf{negl}(\lambda),$$

where $\mathsf{dk}^\star$ *ranges over syntactically valid decoder states for* PRC*.*

Figure 4 illustrates the definition. This immediately rules out decoder openings. A decoder-non-committing opening must output, after corruption, a syntactically valid decoder state $\mathsf{dk}^\star$ that explains the previously sampled dummy codewords. In particular, before requiring robustness on the whole δ-neighborhood, it must at least satisfy $\mathtt{Decode}(\mathsf{dk}^\star, c_i) \neq \perp$ for every previously issued dummy codeword c_i. Targeted low opening capacity says that *no* such valid $\mathsf{dk}^\star$ exists for sufficiently many independent uniform dummy codewords, and hence no decoder-opening mechanism can exist.

Lemma 1 (No Decoder Openings with TLOC). *Let* $\mathsf{PRC} = (\mathtt{KGen}, \mathtt{Encode}, \mathtt{Decode})$ *be a* δ*-robust pseudorandom code. If* PRC *has* $(\mathbf{m}, q)$*-targeted low opening capacity for some* $q \in \mathbb{N}$ *and* $\mathbf{m} \in \Sigma^q$*, then* PRC *does not admit a decoder-non-committing pseudorandom code compatible with* $\mathcal{F}_{\mathsf{PRC}}$ *that satisfies* δ*-opening robustness or* IND-OPEN*.*

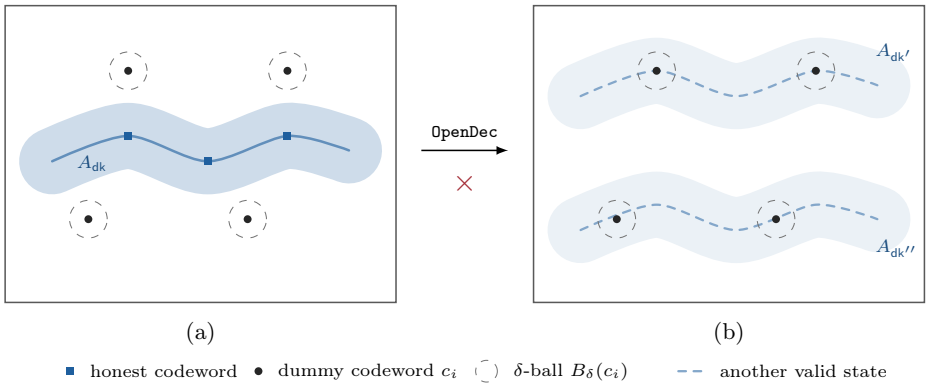

Fig. 4: Targeted low opening capacity as rigid acceptance. In (a), the accepting set A_{dk} of an honest state misses the uniform dummies $c_1, \ldots, c_q$ together with their δ-neighborhoods. In (b), every other valid decoder state $\mathsf{dk}', \mathsf{dk}'', \ldots$ yields an accepting set of the same shape, only moved. A moved copy may cover a few of the δ-balls, but with overwhelming probability none covers all of them, so $\mathsf{NC\text{-}PRC}.\mathtt{OpenDec}(\cdot)$ has no state to output.

Proof (Sketch). Assume, toward contradiction, that PRC admits a compatible decoder-non-committing pseudorandom code $\mathsf{NC\text{-}PRC} = (\mathtt{SimKGen}, \mathtt{SimCode}, \mathtt{OpenDec})$. Run $\mathsf{NC\text{-}PRC}.\mathtt{SimKGen}(1^\lambda)$, and for each m_i in the message sequence $\mathbf{m} = (m_1, \ldots, m_q)$ invoke $\mathsf{NC\text{-}PRC}.\mathtt{SimCode}(\mathsf{pp})$ to obtain a dummy codeword c_i and opening trapdoor τ_{c_i}, labeling it with m_i, forming the opening transcript $\mathcal{T} = \{(m_i, c_i, \tau_{c_i})\}_{i\in[q]}$ and compute $\mathsf{dk}^\star \leftarrow \mathsf{NC\text{-}PRC}.\mathtt{OpenDec}(\mathsf{st}, \tau_k, \mathcal{T})$.

By δ-opening robustness, the opened decoder state must explain the dummy transcript. In particular, except with negligible probability, each codeword c_i must decode to its prescribed message m_i under $\mathsf{dk}^\star$. Thus $\mathtt{Decode}(\mathsf{dk}^\star, c_i) = m_i$ for every $i \in [q]$. This gives a syntactically valid decoder state opening q independent uniform strings to the target sequence $\mathbf{m}$, contradicting $(\mathbf{m}, q)$-targeted low opening capacity. For IND-OPEN, consider the adversary that queries $m_1, \ldots, m_q$, corrupts, and outputs 1 iff the revealed state decodes every c_i to m_i. In the real branch this holds except with negligible probability by δ-robustness at distance 0, while in the simulated branch the dummy codewords are independent uniform strings, so targeted low opening capacity makes it fail except with negligible probability. □

We defer the full proof of Lemma 1 to Appendix C.4.

We emphasize that low opening capacity is only a barrier condition, not a characterization: its failure merely means some syntactically valid decoder state may *pointwise* explain the dummy codewords, whereas a decoder-non-committing code must further provide an indistinguishable corruption view and, in the secret-key case with $\mathsf{ek} = \mathsf{dk}$, δ-neighborhood consistency. Hence any code satisfying TLOC cannot support UC-secure adaptive openings.

A Gap between Soundness and Decoder Openings. Since $\mathcal{F}_{\mathsf{PRC}}$ samples dummy codewords $c \leftarrow\!\!\$\ \Sigma^\star$ uniformly, it is more natural to consider the following random-string form of soundness instead of the ordinary soundness with a fixed codeword.

Definition 14 (Random-String Soundness). *A pseudorandom code* $\mathsf{PRC} = (\mathtt{KGen}, \mathtt{Encode}, \mathtt{Decode})$ *satisfies random-string soundness if*

$$\Pr_{\substack{(\mathsf{ek},\mathsf{dk})\leftarrow\mathtt{KGen}(1^\lambda)\\ c\leftarrow\!\!\$\,\Sigma^\star}}\left[\mathtt{Decode}(\mathsf{dk}, c) \neq \bot\right] \leq \mathsf{negl}(\lambda).$$

Random-string soundness says that a uniformly sampled codeword is unlikely to decode under an independently generated honest decoder state. In contrast, targeted low opening capacity rules out a stronger post-hoc explanation: after many uniform dummy strings are fixed, no syntactically valid decoder state can exist to make them decode to a prescribed message sequence.

Proposition 1 (TLOC implies Random-String Soundness). *Let* $\mathsf{PRC} = (\mathtt{KGen}, \mathtt{Encode}, \mathtt{Decode})$ *be a pseudorandom code with message space* Σ *and codeword space* $\Sigma^\star$*, and assume* $|\Sigma| = \mathsf{poly}(\lambda)$*. Suppose that, for every message* $m \in \Sigma$*,* PRC *has* $((m), 1)$*-targeted low opening capacity. Then* PRC *satisfies random-string soundness.*

Proof. We prove the contrapositive. Suppose that random-string soundness fails. Since $|\Sigma| = \mathsf{poly}(\lambda)$, there exists a fixed message $m^\star \in \Sigma$ such that

$$\Pr_{\substack{(\mathsf{ek},\mathsf{dk})\leftarrow\mathtt{KGen}(1^\lambda)\\ c\leftarrow\!\!\$\,\Sigma^\star}}\left[\mathtt{Decode}(\mathsf{dk}, c) = m^\star\right]$$

is non-negligible. Since an honestly generated decoder state dk is syntactically valid, $\Pr_{c\leftarrow\!\!\$\,\Sigma^\star}[\exists\, \mathsf{dk}^\star : \mathtt{Decode}(\mathsf{dk}^\star, c) = m^\star]$ is non-negligible. Therefore PRC does not have $((m^\star), 1)$-targeted low opening capacity. □

This, in turn, suggests that UC-secure PRCs against adaptive corruption require a *gap* between random-string soundness and targeted low opening capacity. A candidate construction should remain (random-string) sound for honestly generated decoder states, while its decoder family should still have enough structured opening capacity to explain dummy codewords after adaptive decoder corruption.

The LDPC-based PRCs of [CG24, AAC+25] do not lie in this desired gap: they are sound, but they also satisfy targeted low opening capacity, as we show in Section 5.3. Consequently, these constructions cannot support decoder-non-committing openings, and thus do not yield a UC-secure PRC against adaptive corruption.

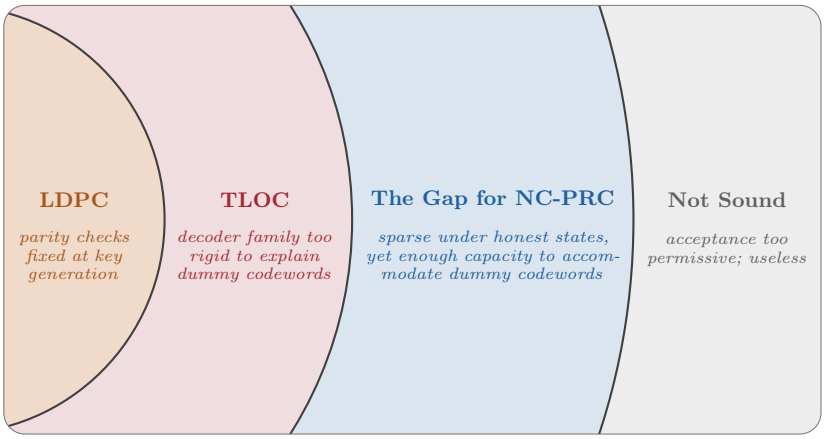

Fig. 5: An illustration of the soundness–opening gap, read as a spectrum of decoder rigidity. On the right, acceptance is too permissive, so uniform dummies already decode under honest states and the code is useless. On the left, in an LDPC-based PRC, the accepting set is fixed at key generation, which falls under our characterization of targeted low opening capacity (Definition 13) and rules out decoder-non-committing codes. A UC-secure PRC against adaptive corruption, i.e., one that admits an NC-PRC, hence must lie in the region in between, sparse enough to reject uniform strings under honest states yet retaining enough opening capacity to explain the dummies after corruption.

5.3 Targeted Low Opening Capacity of LDPC-based PRC

We next instantiate this barrier for the LDPC-based PRC of [CG24] (recalled in Appendix A). Its decoder state contains a sparse parity-check matrix, which gives the decoder family targeted low opening capacity. Hence the required decoder opening does not exist for this instantiation. Therefore, a natural protocol Π_{PRC} based on LDPC cannot realize $\mathcal{F}_{\mathsf{PRC}}$ under adaptive decoder corruption.

On Zero-Bit LDPC-Based PRCs. In Lemma 2, we first justify this for the zero-bit LDPC-based PRC in Construction 5, which only has message space $\Sigma = \{1\}$. Therefore, all the messages in $\mathbf{m}$ are fixed and equal 1.

Lemma 2. *Let* $\mathsf{LDPC\text{-}PRC}_0[n, d, t, r, \eta, \zeta]$ *be the zero-bit LDPC-based PRC in Construction 5. If*

$$q = \omega\left(\frac{r \log \binom{n}{t} + r + \log \lambda}{\log(1 + 2\zeta)}\right),$$

then $\mathsf{LDPC\text{-}PRC}_0$ *satisfies* $(\mathbf{m}, q)$*-targeted low opening capacity for* $\mathbf{m} \in \{1\}^q$.

Proof. Let $c_1, \ldots, c_q \leftarrow\!\!\$\ \{0,1\}^n$ be independent uniform strings. We bound the probability that there exists a syntactically valid decoder state (H, z) such that all c_i are accepted, namely $\forall i \in [q]$, $\mathsf{wt}(H(c_i \oplus z)) < \alpha r$ for $\alpha := 1/2 - \zeta$. Fix any $H \in \mathcal{S}_{t,r,n}$ and any $z \in \{0,1\}^n$. For a uniform c, define $y := \mathsf{wt}(H(c \oplus z))$. Since

every row of H is nonzero and $c \oplus z$ is uniform, each coordinate of $H(c \oplus z)$ is an unbiased bit. Hence $\mathbb{E}[y] = r/2$. Let $p_{H,z} := \Pr[\mathsf{wt}(H(c \oplus z)) < \alpha r]$. Using only the bound $y < r$, we have

$$\mathbb{E}[y] = \frac{r}{2} < \alpha r \cdot p_{H,z} + r \cdot (1 - p_{H,z}).$$

Therefore

$$p_{H,z} < \frac{1}{2(1-\alpha)} = \frac{1}{1+2\zeta}.$$

Thus, for fixed (H, z),

$$\Pr\left[\forall i \in [q],\ \mathsf{wt}(H(c_i \oplus z)) < \alpha r\right] \leq (1+2\zeta)^{-q}.$$

It remains to union bound over all syntactically valid decoder states. There are at most $\binom{n}{t}^r$ possible ordered t-sparse parity-check matrices H. Moreover, for fixed H, the value of z affects the predicate only through the syndrome shift $Hz \in \{0,1\}^r$, so there are at most 2^r relevant choices of z. Hence

$$\Pr\left[\exists (H, z) : \forall i \in [q],\ \mathsf{wt}(H(c_i \oplus z)) < \alpha r\right] \leq \binom{n}{t}^r \cdot 2^r \cdot (1+2\zeta)^{-q}.$$

By the assumed lower bound on q, this quantity is negligible in λ. Therefore, except with negligible probability, no syntactically valid decoder state accepts all q independent uniform strings. This is exactly low opening capacity for $\mathsf{LDPC\text{-}PRC}_0$. □

Lifting to One-Bit LDPC-Based PRCs. Lemma 2 naturally extends to the single-bit LDPC-based PRC with $\Sigma = \{0,1\}$ in Construction 6, which essentially runs two instances of $\mathsf{LDPC\text{-}PRC}_0$ for $m = 0$ and $m = 1$. We formalize this in Corollary 1.

Corollary 1. *Let* $\mathsf{LDPC\text{-}PRC}_1[n, d, t, r, \eta, \zeta]$ *be the single-bit LDPC-based PRC from Construction 6, obtained from two copies of* $\mathsf{LDPC\text{-}PRC}_0[n, d, t, r, \eta, \zeta]$. *Suppose that* $\mathsf{LDPC\text{-}PRC}_0$ *has* $((1)_{i=1}^{q}, q)$*-targeted low opening capacity. Then, for every message sequence* $\mathbf{m} = (b_1, \ldots, b_{2q}) \in \{0,1\}^{2q}$, $\mathsf{LDPC\text{-}PRC}_1$ *has* $(\mathbf{m}, 2q)$*-targeted low opening capacity.*

Proof. We prove the contrapositive. Suppose that $\mathsf{LDPC\text{-}PRC}_1$ does not have $(\mathbf{m}, 2q)$-targeted low opening capacity for some fixed $\mathbf{m} = (b_1, \ldots, b_{2q}) \in \{0,1\}^{2q}$. Then, with non-negligible probability over independent uniform strings $c_1, \ldots, c_{2q}$, there exists a syntactically valid one-bit decoder state $\mathsf{dk}^\star = (\mathsf{dk}_0^\star, \mathsf{dk}_1^\star)$ such that $\mathsf{LDPC\text{-}PRC}_1.\mathtt{Decode}(\mathsf{dk}^\star, c_i) = b_i$ for $i \in [2q]$. For $b \in \{0,1\}$, let $I_b := \{i \in [2q] : b_i = b\}$. By pigeonhole, there exists $b^\star \in \{0,1\}$ such that $|I_{b^\star}| \geq q$. For every $i \in I_{b^\star}$, the fact that $\mathsf{LDPC\text{-}PRC}_1.\mathtt{Decode}(\mathsf{dk}^\star, c_i) = b^\star$ implies, by the definition of the single-bit decoder, that the $b^\star$-th underlying zero-bit detector accepts $\mathsf{LDPC\text{-}PRC}_0.\mathtt{Decode}(\mathsf{dk}_{b^\star}^\star, c_i) = 1$. Choose any subset $J \subseteq I_{b^\star}$ of size

q. Since the strings $c_1, \ldots, c_{2q}$ are independent uniform strings, the subcollection $\{c_i\}_{i \in J}$ is also a collection of q independent uniform strings. Moreover, $\mathsf{LDPC\text{-}PRC}_0.\mathtt{Decode}(\mathsf{dk}^\star_{b^\star}, c_i) = 1$ for every $i \in J$.

Thus $\mathsf{dk}^\star_{b^\star}$ is a syntactically valid zero-bit decoder state that opens q independent uniform strings to the all-1 message sequence, contradicting the $((1)_{i=1}^q, q)$-targeted low opening capacity of $\mathsf{LDPC\text{-}PRC}_0$. Therefore $\mathsf{LDPC\text{-}PRC}_1$ has $(\mathbf{m}, 2q)$-targeted low opening capacity. □

Lifting to Multi-Bit LDPC-Based PRCs. We next lift the targeted low-opening bound from the single-bit LDPC-based PRC to the multi-bit wrappers. The only additional loss comes from union-bounding over the single-bit transcripts induced by a successful multi-bit opening, and, in the secret-key case, over the syntactically valid coordinate permutations.

Corollary 2. *Let* $\mathsf{LDPC\text{-}PRC}_1[n, d, t, r, \eta, \zeta]$ *be the single-bit LDPC-based PRC. Suppose that for every bit transcript* $\mathbf{b} = (b_{i,j})_{i \in [q], j \in [\ell]} \in \{0,1\}^{q\ell}$, $\mathsf{LDPC\text{-}PRC}_1$ *satisfies* $(\mathbf{b}, q\ell)$*-targeted low opening capacity with failure probability at most* $\varepsilon(\lambda)$*. If* $2^{q\ell}\varepsilon(\lambda) = \mathsf{negl}(\lambda)$*, then* $\mathsf{PRC}^{\mathsf{pk}}_{\mathsf{mb}}$ *has targeted low opening capacity. If* $|\Pi_\lambda| \cdot 2^{q\ell}\varepsilon(\lambda) = \mathsf{negl}(\lambda)$*, then* $\mathsf{PRC}^{\mathsf{sk}}_{\mathsf{mb}}$ *has targeted low opening capacity, where* Π_λ *denotes the set of syntactically valid coordinate permutations in the decoder state of* $\mathsf{PRC}^{\mathsf{sk}}_{\mathsf{mb}}$.

Proof. We first consider the public-key construction. Fix any target message sequence $\mathbf{m} = (m_1, \ldots, m_q)$, and sample independent uniform multi-bit strings $c_i = c_{i,1} \| \cdots \| c_{i,\ell} \| w_i$ for $i \in [q]$. If there exists a syntactically valid decoder state $\mathsf{dk}^\star$ such that $\mathsf{PRC}^{\mathsf{pk}}_{\mathsf{mb}}.\mathtt{Decode}(\mathsf{dk}^\star, c_i) = m_i$ for all $i \in [q]$, then all underlying single-bit decoding calls must succeed. Hence the opening induces a bit transcript $\mathbf{b} = (b_{i,j})_{i \in [q], j \in [\ell]} \in \{0,1\}^{q\ell}$ such that $\mathsf{LDPC\text{-}PRC}_1.\mathtt{Decode}(\mathsf{dk}^\star_1, c_{i,j}) = b_{i,j}$ for all i, j.

For each fixed $\mathbf{b}$, this event has failure probability at most $\varepsilon(\lambda)$ by the targeted low opening capacity of $\mathsf{LDPC\text{-}PRC}_1$. Union-bounding over all $2^{q\ell}$ transcripts gives

$$\Pr\left[\exists\, \mathsf{dk}^\star : \forall i,\ \mathsf{PRC}^{\mathsf{pk}}_{\mathsf{mb}}.\mathtt{Decode}(\mathsf{dk}^\star, c_i) = m_i\right] \le 2^{q\ell}\varepsilon(\lambda) = \mathsf{negl}(\lambda).$$

For the secret-key construction, fix a syntactically valid permutation $\pi \in \Pi_\lambda$. Since the c_i are uniform, so are $\pi^{-1}(c_i)$, and the same argument gives failure probability at most $2^{q\ell}\varepsilon(\lambda)$ for this fixed π. A union bound over Π_λ gives

$$\Pr\left[\exists\, \mathsf{dk}^\star : \forall i,\ \mathsf{PRC}^{\mathsf{sk}}_{\mathsf{mb}}.\mathtt{Decode}(\mathsf{dk}^\star, c_i) = m_i\right] \le |\Pi_\lambda| 2^{q\ell}\varepsilon(\lambda) = \mathsf{negl}(\lambda).$$

□

We stress that these are hypotheses on the parameters rather than unconditional claims: they must be checked for a concrete instantiation, and hold whenever the threshold on q in Lemma 2 is exceeded by the additive slack $q\ell + \log|\Pi_\lambda|$ needed to absorb the union bounds.

On Transform to Ideal/CCA Security. The Fujisaki–Okamoto (FO) [FO99]-like transform of [AAC+25] (illustrated in Fig. 7 in Appendix A) is applied on top of the secret-key or public-key multi-bit PRC. Hence any decoder opening for the transformed construction would in particular induce a decoder opening for the underlying multi-bit PRC. Therefore, the same targeted low-opening obstruction from Corollary 2 applies.

Proposition 2. *Neither the public-key nor the secret-key variant of the protocol* Π_{PRC} *in Fig. 7 realizes* $\mathcal{F}_{\mathsf{PRC}}$ *against an adversary that adaptively corrupts parties when* $\overline{\mathsf{PRC}}$ *is instantiated with* $\mathsf{PRC}^{\mathsf{pk}}_{\mathsf{mb}}$ *or* $\mathsf{PRC}^{\mathsf{sk}}_{\mathsf{mb}}$ *assuming the parameter conditions of Corollary 2 hold for* $\mathsf{PRC}^{\mathsf{pk}}_{\mathsf{mb}}$ *and* $\mathsf{PRC}^{\mathsf{sk}}_{\mathsf{mb}}$.

Proof. The proof follows Lemma 1, Corollary 2, and Theorem 3. For the public-key variant, Theorem 3 covers ideal-process adversaries that generate the key pair with `KGen` and answer fresh queries with the real decoder. An arbitrary ideal-process adversary still induces $\mathsf{NC\text{-}PRC}.\{\mathtt{SimKGen}, \mathtt{SimCode}, \mathtt{OpenDec}\}$ satisfying IND-OPEN against adversaries making no decoding queries, as noted after Theorem 3, and the adversary in the proof of Lemma 1 makes none. Hence no ideal-process adversary exists. □

On Other PRC Constructions. The counting behind Lemma 2 is not specific to parity checks. Consider any PRC whose honest decoder states have length at most $K = K(\lambda)$, fixed at key generation, as in all constructions we are aware of [CG24, GM24, GG24, AAC+25, CGG+25, CMY26, DJK+26]. After q honest encodings and an adaptive decoder corruption, $\mathcal{Z}$ can test the revealed state $\mathsf{dk}^\star$: it has length at most K, it accepts every issued codeword, and it accepts a fresh uniform string with probability at most $1/2$. An honest dk passes all three tests except with negligible probability, by correctness and by random-string soundness (Definition 14), which follows from soundness. A fixed state passing the last test accepts q independent uniform strings with probability at most 2^{-q}, so a union bound over the at most 2^{K+1} candidate states shows that for $q \geq K + \omega(\log \lambda)$, no state passing all three tests explains the dummy codewords, except with negligible probability. Hence no such PRC realizes $\mathcal{F}_{\mathsf{PRC}}$ under adaptive decoder corruption against environments issuing an unbounded number of codewords, whatever its decoding structure. This is the PRC analogue of the lower bound of [Nie02]: the decoder state must grow with the number of codewords to be opened, which is why the construction of Section 6.1 fixes an a-priori bound Q on openings and pads its circuits to size $s(\lambda, Q)$. The question for a concrete PRC is therefore whether it admits openings of a bounded number Q of dummy codewords, which we discuss next.

Decoder Openings for Known PRCs. Table 2 sketches the answer for known constructions, grouped by what determines acceptance under a decoder state.

First, when acceptance is fixed by the public key, as in the public-key PRCs of [DFV26, DJK+26] and the planted-hyperloop construction of [GG24], the decoder accepts only words close to a structure determined by ek, which is

published before any dummy codeword is drawn. A uniform dummy is far from that structure except with negligible probability, so it is rejected under every decoder state consistent with ek, and no opening exists even for $Q = 1$.

Second, when the secret is short or sparse, there are too few decoder states to explain many dummy codewords. This is targeted low opening capacity for the LDPC-based PRCs of [CG24, AAC+25], and the same counting applies to the sparse checks of the space-bounded construction of [GG24], to the binary permuted codes of [CGG+25], whose key is one of only $n!$ permutations, and to the symbol-wise PRP of [DMR25], whose λ-bit key cannot explain even one dummy codeword of length $n \gg \lambda$.

Third, when the key contains tables that can be reprogrammed after the dummy codewords are drawn, a bounded opening is possible. The mask of the local weak PRF construction of [GM24] can be reprogrammed for one dummy codeword, the per-symbol permutations of the large-alphabet permuted codes of [CGG+25] can be reprogrammed as long as $Q^2 n \ll |\Sigma|$, and an iO-programmed PRP in the construction of [DMR25] opens any bounded Q, heuristically as in our construction.

We emphasize that these are sketches based on the structure of each decoder, not theorems, and that the thresholds depend on parameters we do not fix. We leave a formal analysis of these cases to future work.

Table 2: Whether known PRCs admit a decoder opening of Q dummy codewords (sketches, not theorems).

Decoder Structure	Construction	Key	Openable Dummies
Acceptance fixed by the public key	Distorted McEliece [DFV26]	PK	no $Q \geq 1$
	Noisy McEliece [DJK+26]	PK	no $Q \geq 1$
	Planted hyperloop [GG24]	PK	no $Q \geq 1$
Short or sparse secret	LDPC [CG24, AAC+25]	PK/SK	no large Q (TLOC)
	Space-bounded [GG24]	SK	no large Q
	Permuted, binary [CGG+25]	SK	no $Q = \omega(\log n)$
	Symbol-wise PRP [DMR25]	SK	no $Q \geq 1$
Programmable tables in the key	Local weak PRF [GM24]	SK	$Q = 1$
	Permuted, large Σ [CGG+25]	SK	$Q^2 n \ll \|\Sigma\|$
	iO-programmed PRP [DMR25]	SK	bounded Q (heuristic)

On Patching LDPC-Based PRC with Multiple Offsets. One may attempt to evade the barrier generically by enlarging the decoder state of Construction 5: keep H untouched, store ℓ independent offsets $z_1, \ldots, z_\ell \leftarrow\$ \mathbb{F}_2^n$, and accept x iff $\exists j \in [\ell]$ with $\mathsf{wt}(H(x \oplus z_j)) \leq \alpha r$. An opening is then immediate, since for each dummy c_i an unused offset can be reassigned as $z_{j_i} := c_i \oplus Gu_i \oplus e_i$ for fresh u_i, e_i, after which c_i decodes via z_{j_i} exactly as an honest codeword would. This does not contradict Lemma 1, whose union bound over decoder states no longer closes once the state space grows by the factor $2^{r\ell}$.

The patch is nevertheless unsatisfying for three reasons. First, the number of openable dummies is bounded by ℓ, which must be fixed at key generation, yielding only a bounded-opening notion. Second, the decoder state grows linearly in this bound, echoing the lower bound of [Nie02] that non-committing schemes without erasures require state proportional to the volume to be opened. Third, IND-OPEN requires care: if honest encoding always uses a designated offset, the environment distinguishes by observing that all honest codewords match a *common* offset while opened dummies match pairwise *distinct* ones, so honest encoding must itself sample $j \leftarrow\!\!\$\ [\ell]$ per codeword and ℓ must be large enough that offset collisions are negligible in both worlds. The accepting fraction also degrades by a factor of ℓ, though it remains negligible for $\ell = \mathsf{poly}(\lambda)$.

6 Non-Committing PRC via Programmable Acceptance

The barrier of Lemma 1 arises because the LDPC decoder family is too rigid: the accepting set of a state (H, z) is fixed in advance, so no valid (H, z) explains q later-drawn uniform strings. Both constructions below instead compile error-correcting codes to make codeword acceptance *programmable*, absorbing dummies into a redundant region apart from honest codewords, as illustrated in Fig. 6. The secret-key one (Section 6.1) uses an obfuscated program, re-obfuscated after the dummies are drawn, and the public-key one (Section 6.2) uses a smooth projective hash whose key retains, given the projection key, enough residual freedom to be opened consistently. We prove the security of both compilers when the coding layer is idealized. In this sense the constructions are heuristic, and a standard-model instantiation of the coding layer is open.

6.1 Secret-Key NC-PRC from Puncturable PRF

We first describe in Construction 1 a zero-bit (i.e., the message space is $\{1\}$) secret-key PRC from a *puncturable PRF* (Definition 16 in Appendix B) [BW13] and an *indistinguishability obfuscator* (iO, Definition 17 in Appendix B) [BGI+01], inspired by [SW14], together with an error-correcting code.

A codeword carries a payload $v \leftarrow\!\!\$\ \mathcal{V}$, sampled uniformly per encoding as a randomizer, which spreads codewords over the exponentially sparse accepting set $\{(v, P(v)) : v \in \mathcal{V}\} \subseteq \Sigma$, and a tag $t \leftarrow P(v)$, where $P := \mathsf{iO}(\mathsf{pPRF}.\mathtt{Eval}(\mathsf{msk}, \cdot))$ is the obfuscated evaluation program with the master key msk hardwired.

Decoding accepts iff the tag recovered by ECC matches P on the recovered payload, so that acceptance is decided by P rather than by a fixed state. Equivocality is then obtained by re-obfuscating P a posteriori with the dummy points programmed in.

Construction 1 (Zero-Bit Secret-Key PRC)**.** *Let* iO *be an indistinguishability obfuscator, let* $\mathsf{pPRF} = (\mathtt{Setup}, \mathtt{Puncture}, \mathtt{Eval}, \mathtt{PuncEval})$ *be a puncturable PRF with domain* $\mathcal{V}$ *and range* $\mathcal{T}$*. Let* $\mathsf{ECC} = (\mathtt{Encode}, \mathtt{Decode})$ *be an error-correcting code with input space* $\Sigma = \mathcal{V} \times \mathcal{T}$*, codeword space* $\Sigma^\star$*, and decoding radius* δ*.*

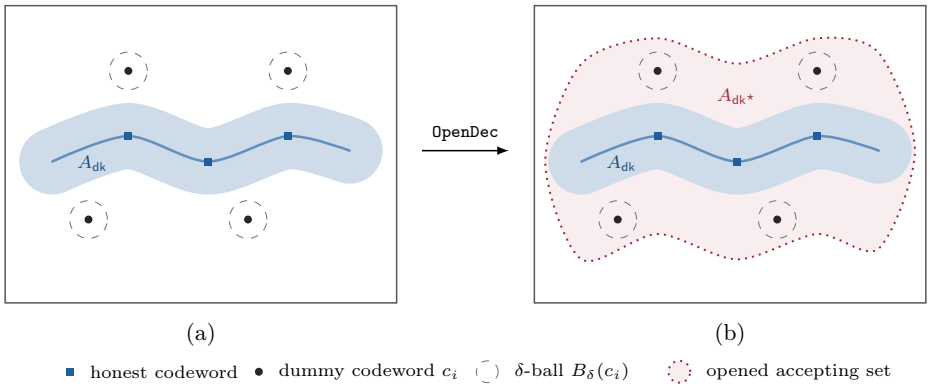

Fig. 6: Programmable acceptance. The blue shaded band is the accepting set $A_{\mathsf{dk}} := \{x \in \Sigma^\star : \mathtt{Decode}(\mathsf{dk}, x) \neq \perp\}$ of an honest state, which occupies a negligible fraction of the codeword space by random-string soundness. In (a), it is fixed at key generation, so the uniform dummies $c_1, \ldots, c_q$ drawn afterwards fall outside it together with their δ-neighborhoods; this is targeted low opening capacity, which precludes any decoder opening. In (b), acceptance is programmable, and $\mathsf{NC\text{-}PRC}.\mathtt{OpenDec}(\cdot)$ programs each c_i into the opened state, enlarging the accepting set along the shape of the honest one to the red shaded region $A_{\mathsf{dk}^\star}$, so that $\bigcup_{i\in[q]} B_\delta(c_i) \subseteq A_{\mathsf{dk}^\star}$.

Let χ be a noise distribution over $\Sigma^\star$ supported within the decoding radius δ. A zero-bit secret-key pseudorandom code $\mathsf{PRC}_0^{\mathsf{sk}}$ is defined as follows.

- $\mathtt{KGen}(1^\lambda)$*: Generate* $\mathsf{msk} \leftarrow \mathsf{pPRF}.\mathtt{Setup}(1^\lambda)$*. Generate an obfuscated program* $P := \mathsf{iO}\big(\mathsf{pPRF}.\mathtt{Eval}(\mathsf{msk}, \cdot)\big)$ *and set* $\mathsf{ek} := \mathsf{dk} := P$*.*
- $\mathtt{Encode}(\mathsf{ek}, 1)$*: Sample* $v \leftarrow\!\!\$\ \mathcal{V}$ *and noise* $e \leftarrow\!\!\$\ \chi$*. Set* $t \leftarrow P(v)$*, and output* $c \leftarrow \mathsf{ECC}.\mathtt{Encode}(v\|t) \oplus e$*.*
- $\mathtt{Decode}(\mathsf{dk}, x)$*: Compute* $v\|t \leftarrow \mathsf{ECC}.\mathtt{Decode}(x)$*, outputting* $\perp$ *if* ECC *fails. If* $t = P(v)$*, output* 1*; otherwise output* $\perp$*.*

For a dummy codeword c_i to be accepted, it suffices to program its decoded payload v_i to its decoded tag t_i for each $i \in [q]$. To this end, we puncture msk at the programmed points $\{v_i\}_{i\in[q]}$ and obfuscate a program $\widehat{P}$ that returns t_i on these points and runs $\mathsf{pPRF}.\mathtt{PuncEval}(\mathsf{psk}, \cdot)$ elsewhere, which agrees with $\mathsf{pPRF}.\mathtt{Eval}(\mathsf{msk}, \cdot)$ outside the punctured set. Since the dummy tags t_i are uniform and pPRF security makes the true values $\mathsf{pPRF}.\mathtt{Eval}(\mathsf{msk}, v_i)$ pseudorandom given psk, the programmed $\widehat{P}$ is indistinguishable from $\mathsf{pPRF}.\mathtt{Eval}(\mathsf{msk}, \cdot)$ when obfuscated.

In Construction 2, we describe the NC-PRC admitted by $\mathsf{PRC}_0^{\mathsf{sk}}$ in Construction 1. For simplicity, we omit the trapdoor τ from the algorithms since it does not contain cryptographic content as discussed in Section 5.1.

Construction 2 (Zero-Bit Secret-Key NC-PRC)**.** *Let* $\mathsf{PRC}_0^{\mathsf{sk}}$ *be the zero-bit secret-key PRC of Construction 1, with a puncturable PRF* $\mathsf{pPRF} = (\mathtt{Setup}, \mathtt{Puncture},$

$\mathtt{Eval}, \mathtt{PuncEval}$) *over domain* $\mathcal{V}$ *and range* $\mathcal{T}$*, an indistinguishability obfuscator* iO*, and an error-correcting code* ECC *with decoding radius* δ*. The decoder-non-committing code* $\mathsf{NC\text{-}PRC}_0^{\mathsf{sk}} = (\mathtt{SimKGen}, \mathtt{SimCode}, \mathtt{OpenDec})$ *is defined as follows.*

- $\mathtt{SimKGen}(1^\lambda)$*: Generate* $\mathsf{msk} \leftarrow \mathsf{pPRF.Setup}(1^\lambda)$*. Set* $\mathsf{pp} := \bot$*,* $\mathsf{st} := \mathsf{msk}$*. Output* $(\mathsf{pp}, \mathsf{st})$*.*
- $\mathtt{SimCode}(\mathsf{pp})$*: Sample* $c \leftarrow\!\!\$\ \Sigma^\star$ *uniformly. Output* c*.*
- $\mathtt{OpenDec}(\mathsf{st}, \mathcal{T})$*: Parse* $\mathsf{st} = \mathsf{msk}$ *and* $\mathcal{T} = \{(m_i, c_i)\}_{i=1}^q$ *with* $m_i = 1$ *for* $i \in [q]$*. For each* $i \in [q]$*, recover* $v_i \| t_i \leftarrow \mathsf{ECC.Decode}(c_i)$*, skipping* c_i *if* ECC *fails. Compute* $\mathsf{psk} \leftarrow \mathsf{pPRF.Puncture}(\mathsf{msk}, \{v_i\}_{i\in[q]})$ *and set* $P^\star := \mathsf{iO}(\widehat{P})$ *where*

$$\widehat{P}(v) := \begin{cases} t_i & \text{if } v = v_i \text{ for some } i \in [q], \\ \mathsf{pPRF.PuncEval}(\mathsf{psk}, v) & \text{otherwise.} \end{cases}$$

Output $\mathsf{dk}^\star := P^\star$*.*

Proposition 3. *Let* iO *be perfectly correct. Assume that, except with negligible probability over* $c \leftarrow\!\!\$\ \Sigma^\star$*,* $\mathsf{ECC.Decode}(c) = v \| t \neq \bot$ *and* $\mathsf{ECC.Decode}(x) = v \| t$ *for every* x *with* $\mathsf{dist}(x, c) < \delta$*, and that the payloads* $v_1, \ldots, v_q$ *decoded from* q *independent uniform strings* $c_1, \ldots, c_q$ *are pairwise distinct except with negligible probability. Then* $\mathsf{NC\text{-}PRC}_0^{\mathsf{sk}}$ *of Construction 2 satisfies* δ*-opening robustness.*

Proof. For the dummy codewords c_i output by $\mathsf{NC\text{-}PRC}_0^{\mathsf{sk}}.\mathtt{SimCode}(\cdot)$, define Bad as the event that some c_i fails to decode to some $v_i \| t_i \neq \bot$ with $\mathsf{ECC.Decode}(x) = v_i \| t_i$ for all $\mathsf{dist}(x, c_i) < \delta$, or that the payloads $v_1, \ldots, v_q$ are not pairwise distinct. By hypothesis and a union bound, $\Pr[\mathsf{Bad}] \leq \mathsf{negl}(\lambda)$.

Conditioned on $\neg\mathsf{Bad}$, the map $v_i \mapsto t_i$ is a well-defined partial function, so $\widehat{P}$ is well defined, and perfect correctness of iO gives $P^\star(v) = \widehat{P}(v)$ for all $v \in \mathcal{V}$. If $\mathcal{A}_2$ outputs x with a unique $(m, c_i) \in \mathsf{SimNear}_{\mathcal{T}}^{\delta}(x)$, then $\mathsf{dist}(x, c_i) < \delta$ forces $\mathsf{ECC.Decode}(x) = v_i \| t_i$ and $P^\star(v_i) = t_i$, whence $\mathtt{Decode}(\mathsf{dk}^\star, x) = 1 = m$; the violation event is thus contained in Bad. □

Proposition 4. *Let all obfuscated circuits in* $\mathsf{PRC}_0^{\mathsf{sk}}$ *and* $\mathsf{NC\text{-}PRC}_0^{\mathsf{sk}}$ *be padded to a common size* $s(\lambda, Q)$ *for an a-priori bound* Q *on openings. Assume that* pPRF *has perfect punctured correctness and pseudorandomness at punctured points,* iO *satisfies indistinguishability,* $|\mathcal{V}|$ *is superpolynomial, and the distance between* $\mathsf{ECC.Encode}(w) \oplus e$ *for* $w \leftarrow\!\!\$\ \Sigma$*,* $e \leftarrow\!\!\$\ \chi$ *and a uniform* $c \leftarrow\!\!\$\ \Sigma^\star$ *is within* $\epsilon(\lambda)$*. Then for every PPT adversary* $\mathcal{A}$ *making at most* $q \leq Q$ *encoding queries, all prior to its corruption query, there exist PPT adversaries* $\mathcal{B}_1, \mathcal{B}_2$ *such that*

$$\mathbf{Adv}^{\text{IND-OPEN}}_{\mathsf{PRC}_0^{\mathsf{sk}}, \mathsf{NC\text{-}PRC}_0^{\mathsf{sk}}}(\mathcal{A}) \;\leq\; \mathbf{Adv}^{\text{IND}}_{\mathsf{iO}}(\mathcal{B}_1) + \mathbf{Adv}^{\text{PRND}}_{\mathsf{pPRF}}(\mathcal{B}_2) + \frac{q^2}{2|\mathcal{V}|} + q \cdot \epsilon(\lambda).$$

Proof (Sketch). We first replace the honest program $P = \mathsf{iO}(\mathsf{pPRF.Eval}(\mathsf{msk}, \cdot))$ returned upon corruption by $\mathsf{iO}(\widehat{P})$ for $\widehat{P}$ defined in Construction 2. By perfect punctured correctness the two circuits are functionally identical, so this reduces to iO indistinguishability. We then replace the tags $\mathsf{pPRF.Eval}(\mathsf{msk}, v_i)$

by uniform $t_i \leftarrow\!\!\$\ \mathcal{T}$, both in the codewords $c_i \leftarrow \mathsf{ECC}.\mathtt{Encode}(v_i \| t_i) \oplus e_i$ and at the hardwired points, via pseudorandomness at punctured points; each c_i now encodes an independent uniform pair $(v_i, t_i) \leftarrow\!\!\$\ \Sigma$, and the opened program is a randomized post-processing $G(c_1, \ldots, c_q)$ of the codewords alone. Replacing each $\mathsf{ECC}.\mathtt{Encode}(v_i \| t_i) \oplus e_i$ by a uniform $c_i \leftarrow\!\!\$\ \Sigma^\star$ then costs $q^2/2|\mathcal{V}| + q \cdot \epsilon(\lambda)$, landing exactly in the simulated branch where G coincides with $\mathsf{NC\text{-}PRC}_0^{\mathsf{sk}}.\mathtt{OpenDec}(\cdot)$. □

We defer the full proof of Proposition 4 to Appendix C.5.

Extension to Single/Multi-Bit PRC. Construction 1 extends to a single-bit PRC (i.e., with message space $\{0,1\}$) by running two independent instances pPRF_0 and pPRF_1 with $\mathsf{msk}_b \leftarrow \mathsf{pPRF}_b.\mathtt{Setup}(1^\lambda)$ and $P_b := \mathsf{iO}(\mathsf{pPRF}_b.\mathtt{Eval}(\mathsf{msk}_b, \cdot))$ for $b \in \{0,1\}$, encoding a message b under P_b and decoding to the unique b whose program accepts. Accordingly, $\mathsf{NC\text{-}PRC}.\mathtt{OpenDec}(\cdot)$ programs each dummy pair (v_i, t_i) only into the program of its message: for $b \in \{0,1\}$, it punctures msk_b at $V_b := \{v_i : m_i = b\}$, defines $\widehat{P}_b$ to return t_i on each $v_i \in V_b$ and to run $\mathsf{pPRF}_b.\mathtt{PuncEval}(\mathsf{psk}_b, \cdot)$ elsewhere, and outputs $\mathsf{dk}^\star := (P_0^\star, P_1^\star)$ with $P_b^\star := \mathsf{iO}(\widehat{P}_b)$. A dummy c_i is then accepted by $P_{m_i}^\star$, while $P_{1-m_i}^\star$, being unprogrammed at v_i, returns a pseudorandom tag matching t_i only with negligible probability, so decoding outputs m_i. A single-bit PRC may be further extended to a multi-bit PRC via the transform in [CG24, AAC+25].

More broadly, it would be valuable to import the machinery developed for non-committing encryption, such as lossy or equivocable encryption, to obtain NC-PRCs from general assumptions rather than heuristically, which we leave open.

Idealization of Coding Layer. Our constructions isolate equivocation in the programmable layer and treat the coding layer as an *idealized* component. At the least, ECC must not itself reject uniform strings: if $\mathsf{ECC}.\mathtt{Decode}$ outputs $\perp$ on a dummy codeword, then $\mathsf{NC\text{-}PRC}.\mathtt{OpenDec}(\cdot)$ has nothing to program, and the opened state rejects the dummy together with its δ-neighborhood. This calls for a *total* decoder, as given by nearest-codeword decoding, e.g., for Reed–Solomon codes [RS60], lattice quantization [CS82], or convolutional codes under Viterbi decoding [Vit67], so that rejection is decided solely by the tag or hash check. Random-string soundness is unaffected, since a uniform string decodes to an unprogrammed pair that passes the check only with negligible probability. This contrasts with the LDPC-based decoder of Construction 5, which rejects at the coding layer itself; our constructions defer that sparsity to the programmable layer, which is precisely what leaves room for opening.

Totality is necessary but not sufficient. The full hypotheses on ECC, namely δ-interiority and closeness of noisy encodings to uniform, together with δ-tightness for the decoding queries of the public-key construction (Proposition 6), are demanding when ECC is public and keyless, for any alphabet size. Indeed, if e is small, $\mathsf{ECC}.\mathtt{Encode}(v \| t) \oplus e$ stays close to the public code, which a uniform string does not, so pseudorandomness fails; if e spreads over the decoding region, the encoding sits near its boundary, as a uniform string does, and a single further

error moves it to a neighboring region, so robustness fails. A robust instantiation therefore calls for secret coding structure. For binary alphabets, this is consistent with the black-box separations of [GGW25, DMR25]. Over alphabets of size 2^λ, however, [DMR25] obtains such structure from one-way functions alone, via a construction close to ours: a Reed–Solomon code applied to a pseudorandom payload, followed by a keyed permutation applied symbol-wise. Adding such a permutation layer on top of ECC would remove the coding-layer hypotheses in this regime, with equivocation obtained by programming the inverted permutation at the symbols of the dummy codewords (e.g., by obfuscating it with iO, as for P). We leave this variant to future work.

On the other hand, all hypotheses hold when ECC is idealized as a stateful, programmable oracle, in the spirit of a programmable random oracle. The oracle keeps a table of *centers*: $\mathtt{Encode}(w)$ samples a uniform c, records (c, w), and returns c; $\mathtt{Decode}(x)$ returns the value of a center within distance δ of x if one exists, and otherwise records x as a new center with a fresh $w' \leftarrow\!\!\$\ \Sigma$ and returns w'. This oracle is a keyless, programmable counterpart of the random PRC model of [DKRR26], a random codebook with Hamming-ball decoding under a secret key, used there as perfectly secure candidates in an impossibility result. Ours idealizes less, as it only provides robust decoding and leaves acceptance to the programmable layer, but it must be programmable by the simulator. Our compilers are sound in this idealized model, and we hence view the constructions as showing that programmable acceptance suffices for equivocation, leaving a robust instantiation of the coding layer open.

6.2 Public-Key NC-PRC from Smooth Projective Hash

In Construction 3, we describe a public-key non-committing PRC from a *smooth projective hash function* (SPHF, Definition 18 in Appendix B) [CS02] over sets $L \subseteq U$ where $L = \{u \in U : \exists\, w \in W,\ (u, w) \in \mathcal{R}\}$ for a witness set W and a relation $\mathcal{R}$. *Projectivity* guarantees that $\mathtt{Hash}(\mathsf{hk}, u)$ under a secret hashing key hk can be computed publicly as $\mathtt{Proj}(\mathsf{hp}, u, w)$ from a projection key hp. Hence the encoding is to sample $(u, w) \in \mathcal{R}$ and run $\mathtt{Proj}(\mathsf{hp}, u, w)$ while the decoding is to check whether the hash matches with $\mathtt{Hash}(\mathsf{hk}, u)$.

Construction 3 (Zero-Bit Public-Key PRC)**.** *Let $L \subseteq U$ be sets, and let $\mathcal{R} \subseteq U \times W$ be a relation for a witness set W such that $L = \{u \in U : \exists\, w \in W,\ (u, w) \in \mathcal{R}\}$. Let $\mathsf{SPHF} = (\mathtt{HashKG}, \mathtt{ProjKG}, \mathtt{Hash}, \mathtt{Proj})$ be a q-smooth projective hash family over (U, L) with hash space $\mathcal{H}$. Let $\mathsf{ECC} = (\mathtt{Encode}, \mathtt{Decode})$ be an error-correcting code with input space $\Sigma = U \times \mathcal{H}$, codeword space $\Sigma^\star$, and decoding radius δ. Let χ be a noise distribution over $\Sigma^\star$ supported within the decoding radius δ. A zero-bit public-key pseudorandom code $\mathsf{PRC}_0^{\mathsf{pk}}$ is defined as follows.*

- $\mathtt{KGen}(1^\lambda)$*: Sample* $\mathsf{hk} \leftarrow \mathsf{SPHF}.\mathtt{HashKG}(1^\lambda)$ *and* $\mathsf{hp} \leftarrow \mathsf{SPHF}.\mathtt{ProjKG}(\mathsf{hk})$*. Output* $\mathsf{ek} := \mathsf{hp}$ *and* $\mathsf{dk} := \mathsf{hk}$*.*
- $\mathtt{Encode}(\mathsf{ek}, 1)$*: Sample* $(u, w) \leftarrow\!\!\$\ U \times W$ *such that* $(u, w) \in \mathcal{R}$ *and noise* $e \leftarrow\!\!\$\ \chi$*. Set* $h \leftarrow \mathsf{SPHF}.\mathtt{Proj}(\mathsf{hp}, u, w)$ *and output* $c \leftarrow \mathsf{ECC}.\mathtt{Encode}(u \| h) \oplus e$*.*

– $\mathtt{Decode}(\mathsf{dk}, x)$*: Compute* $u\|h \leftarrow \mathsf{ECC}.\mathtt{Decode}(x)$*, outputting* $\perp$ *if* ECC *fails. If* $h = \mathsf{SPHF}.\mathtt{Hash}(\mathsf{hk}, u)$*, output* 1*; otherwise output* $\perp$*.*

All equivocation capacity resides on $U \setminus L$. The projection key hp determines the hash values only on L, while the values $\mathtt{Hash}(\mathsf{hk}, u)$ for $u \in U \setminus L$ remain (close to) uniform even conditioned on hp, so many hashing keys are consistent with a single hp. Since a dummy codeword decodes to a statement outside L with overwhelming probability, $\mathsf{NC\text{-}PRC}_0^{\mathsf{pk}}.\mathtt{OpenDec}(\cdot)$ can exploit exactly this freedom: it samples, a posteriori, a hashing key consistent with both hp and the q dummy codewords.

In Construction 4, we describe the NC-PRC admitted by $\mathsf{PRC}_0^{\mathsf{pk}}$ in Construction 3. As in Construction 2, we omit the trapdoors τ.

Construction 4 (Public-Key Zero-Bit NC-PRC)**.** *Let* $\mathsf{PRC}_0^{\mathsf{pk}}$ *be the zero-bit public-key PRC of Construction 3, with* q*-smooth projective hash family* $\mathsf{SPHF} = (\mathtt{HashKG}, \mathtt{ProjKG}, \mathtt{Hash}, \mathtt{Proj})$ *over* (U, L) *with hash space* $\mathcal{H}$*, and an error-correcting code* ECC *with decoding radius* δ*. The decoder-non-committing code* $\mathsf{NC\text{-}PRC}_0^{\mathsf{pk}} = (\mathtt{SimKGen}, \mathtt{SimCode}, \mathtt{OpenDec})$ *is defined as follows.*

– $\mathtt{SimKGen}(1^\lambda)$*: Sample* $\mathsf{hk} \leftarrow \mathsf{SPHF}.\mathtt{HashKG}(1^\lambda)$ *and* $\mathsf{hp} \leftarrow \mathsf{SPHF}.\mathtt{ProjKG}(\mathsf{hk})$*. Set* $\mathsf{pp} := \mathsf{hp}$*,* $\mathsf{st} := (\mathsf{hk}, \mathsf{hp})$*. Output* $(\mathsf{pp}, \mathsf{st})$*.*

– $\mathtt{SimCode}(\mathsf{pp})$*: Sample* $c \leftarrow\!\!\$\ \Sigma^\star$ *uniformly. Output* c*.*

– $\mathtt{OpenDec}(\mathsf{st}, \mathcal{T})$*: Parse* $\mathsf{st} = (\mathsf{hk}, \mathsf{hp})$ *and* $\mathcal{T} = \{(m_i, c_i)\}_{i=1}^q$ *with* $m_i = 1$ *for* $i \in [q]$*. For each* $i \in [q]$*, recover* $u_i\|h_i \leftarrow \mathsf{ECC}.\mathtt{Decode}(c_i)$*, skipping* c_i *if* ECC *fails or* $u_i \in L$*. Sample* $\mathsf{hk}^\star$ *uniformly from*

$$\mathcal{K}^\star := \left\{ \mathsf{hk}' \,:\, \mathsf{SPHF}.\mathtt{ProjKG}(\mathsf{hk}') = \mathsf{hp} \ \wedge\ \forall i \in [q],\ \mathsf{SPHF}.\mathtt{Hash}(\mathsf{hk}', u_i) = h_i \right\},$$

and output $\mathsf{dk}^\star := \mathsf{hk}^\star$*; if* $\mathcal{K}^\star = \varnothing$*, output* $\mathsf{dk}^\star := \mathsf{hk}$*.*

Proposition 5. *Assume that, except with negligible probability over* $c \leftarrow\!\!\$\ \Sigma^\star$*,* $\mathsf{ECC}.\mathtt{Decode}(c) = u\|h \neq \perp$ *and* $\mathsf{ECC}.\mathtt{Decode}(x) = u\|h$ *for every* x *with* $\mathsf{dist}(x, c) < \delta$*, the statements* $u_1, \ldots, u_q$ *decoded from* q *independent uniform strings* $c_1, \ldots, c_q$ *are pairwise distinct with* $u_i \notin L$ *for all* $i \in [q]$*, except with negligible probability, and that* SPHF *is* q*-smooth. Then* $\mathsf{NC\text{-}PRC}_0^{\mathsf{pk}}$ *of Construction 4 satisfies* δ*-opening robustness.*

Proof. For the dummy codewords c_i output by $\mathsf{NC\text{-}PRC}_0^{\mathsf{pk}}.\mathtt{SimCode}(\cdot)$, define Bad as the event that (i) some c_i fails to decode to some $u_i\|h_i \neq \perp$ with $\mathsf{ECC}.\mathtt{Decode}(x) = u_i\|h_i$ for all $\mathsf{dist}(x, c_i) < \delta$; (ii) the statements $u_1, \ldots, u_q$ are not pairwise distinct or some $u_i \in L$; or (iii) $\mathcal{K}^\star = \varnothing$. Events (i) and (ii) occur with negligible probability by hypothesis, and for (iii), q-smoothness makes $(\mathsf{hp}, \{h_i\}_i)$ statistically close to $(\mathsf{hp}, \{\mathsf{SPHF}.\mathtt{Hash}(\mathsf{hk}, u_i)\}_i)$, under which the honest hk itself lies in $\mathcal{K}^\star$; hence $\Pr[\mathsf{Bad}] \leq \mathsf{negl}(\lambda)$.

Conditioned on $\neg\mathsf{Bad}$, the algorithm $\mathsf{NC\text{-}PRC}_0^{\mathsf{pk}}.\mathtt{OpenDec}(\cdot)$ outputs some $\mathsf{hk}^\star \in \mathcal{K}^\star$ with $\mathsf{SPHF}.\mathtt{Hash}(\mathsf{hk}^\star, u_i) = h_i$ for all $i \in [q]$. If $\mathcal{A}_2$ outputs x with a unique $(m, c_i) \in \mathsf{SimNear}_{\mathcal{T}}^{\delta}(x)$, then $\mathsf{dist}(x, c_i) < \delta$ forces $\mathsf{ECC}.\mathtt{Decode}(x) = u_i\|h_i$ by (i), so $\mathtt{Decode}(\mathsf{hk}^\star, x) = 1 = m$; the violation event is thus contained in Bad. □

Proposition 6. *Assume that* SPHF *satisfies projectivity, sparsity with bound* $\epsilon_{\mathrm{sp}}(\lambda)$*, subset membership indistinguishability, and* $(q+1)$*-smoothness with distance* $\epsilon_{\mathrm{sm}}(\lambda)$*, that* $|U|$ *is superpolynomial, and that the statistical distance between* $\mathsf{ECC.Encode}(w) \oplus e$*, for independent* $w \leftarrow\!\!\$\ \Sigma$ *and* $e \leftarrow\!\!\$\ \chi$*, and a uniform* $c \leftarrow\!\!\$\ \Sigma^\star$ *is at most* $\epsilon(\lambda)$*. Assume moreover* δ-tightness*: for* $c \leftarrow\!\!\$\ \Sigma^\star$*, every PPT algorithm given* c *outputs some* x *with* $\mathsf{dist}(x, c) \geq \delta$ *and* $\mathsf{ECC.Decode}(x) = \mathsf{ECC.Decode}(c) \neq \bot$ *with probability at most* $\epsilon_{\mathrm{tg}}(\lambda)$*. Then for every PPT adversary* $\mathcal{A}$ *making at most* q *encoding queries and at most* Q_d *decoding queries, all prior to its corruption query, there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\text{IND-OPEN}}_{\mathsf{PRC}_0^{\mathsf{pk}},\mathsf{NC\text{-}PRC}_0^{\mathsf{pk}}}(\mathcal{A}) \;\leq\; q \cdot \mathbf{Adv}^{\text{IND-SM}}_{\mathsf{SPHF}}(\mathcal{B}) + \epsilon_{\mathrm{sm}} + q \cdot \epsilon_{\mathrm{sp}} + \frac{q^2}{2|U|} + 2q \cdot \epsilon(\lambda) + q \cdot \epsilon_{\mathrm{tg}} + Q_d \cdot \left(3\epsilon_{\mathrm{sm}} + \frac{2}{|\mathcal{H}|}\right).$$

Proof (Sketch). We first replace each honest statement $u_i \in L$ by a uniform $u_i \leftarrow\!\!\$\ U$, computing the hash as $\mathsf{SPHF.Hash}(\mathsf{hk}, u_i)$, which by projectivity is identical on L and reduces to subset membership indistinguishability (IND-SM). Once all statements lie outside L, q-smoothness makes the joint tuple $(\mathsf{hp}, h_1, \ldots, h_q)$ statistically close to uniform hashes, and uniformity of $\mathtt{HashKG}$ makes the real key, conditioned on this tuple, uniform over the fiber $\mathcal{K}^\star$, which is exactly the distribution of $\mathsf{hk}^\star$. Decoding queries, which concern fresh words only, are answered under the key hk in st before corruption, as the simulated branch does, rather than under $\mathsf{hk}^\star$: the two keys agree on L by projectivity, a fresh word cannot decode to the statement of a dummy codeword by δ-tightness, and on any other statement outside L both keys reject except with probability $3\epsilon_{\mathrm{sm}} + 2/|\mathcal{H}|$ per query by smoothness. Replacing each noisy encoding of the now-uniform pair (u_i, h_i) by a uniform codeword then costs $q \cdot \epsilon(\lambda)$, landing in the simulated branch. □

We defer the proof of Proposition 6 to Appendix C.6.

Remark 1 (On Sampling $\mathsf{dk}^\star$ *Efficiently).* Construction 3 similarly extends to single-bit PRC by running two instances of SPHF and to multi-bit PRC following the transforms in [CG24, AAC+25].

The main obstacle is that uniform sampling from $\mathcal{K}^\star$ must be efficient (and we assume so), and jointly so for all q pairs (u_i, h_i). This should hold when $\mathtt{ProjKG}(\cdot)$ and $\mathtt{Hash}(\cdot, u)$ are linear in hk where the q hash conditions, together with the projection constraint, form a single linear system, so $\mathcal{K}^\star$ is an affine subspace, from which uniform sampling reduces to Gaussian elimination. Nevertheless, the efficiency of $\mathsf{NC\text{-}PRC}_0^{\mathsf{pk}}.\mathtt{OpenDec}(\cdot)$ and a bound for q must be studied with concrete instantiations of SPHF.

7 A Plain-Model Barrier in Public-Key PRC

We finally isolate an obstruction specific to public-key PRCs, *orthogonal* to the adaptive-corruption gap of Section 5, arising already with no corrupted party.

Call a codeword c *fresh* if $\mathsf{Near}^{\delta}_{\mathsf{Issued}[\mathsf{sid}]}(c) = \varnothing$. Since ek is public, anyone can produce a fresh codeword that an honest decoder accepts as m, which the ideal execution must then return to $\mathcal{F}_{\mathsf{PRC}}$; yet m cannot be computed from (ek, c) alone, since any such public extractor breaks pseudorandomness (Lemma 4), so a realization must draw on hidden setup state. This also highlights that PRC is not publicly checkable, separating PRCs from PKE, where this barrier is vacuous and Naor–Yung-style transforms [NY90] with NIZKs even enable CCA security. For PRCs, any publicly verifiable verdict contradicts pseudorandomness, making watermark detection inherently *designated-verifier*.

Lemma 3 (Necessary Fresh Codeword Explanation). *Let Π_{PRC} UC-realize $\mathcal{F}_{\mathsf{PRC}}$. Except with negligible probability, whenever an uncorrupted decoder $\mathcal{P}_{\mathsf{dcd}}$ decodes a fresh codeword c as $m \in \Sigma$ in the real world, the ideal-process adversary $\mathcal{S}_{\mathsf{PRC}}$ must return the same m in the fresh-codeword branch of $\mathcal{F}_{\mathsf{PRC}}$ in the ideal world.*

Proof. If $\mathcal{S}_{\mathsf{PRC}}$ returns $\perp$, the ideal execution aborts while the real execution outputs m. If $\mathcal{S}_{\mathsf{PRC}}$ returns $m' \neq m$, the two executions output different messages. Either case gives an immediate UC distinguisher. Hence $\mathcal{S}_{\mathsf{PRC}}$ must return the same m. □

The explanation is trivial in secret-key PRCs: a fresh valid codeword is infeasible to generate until corruption discloses the secret key $\mathsf{ek} = \mathsf{dk}$, after which $\mathcal{S}_{\mathsf{PRC}}$ simply runs the real decoder on fresh inputs. In the public-key case, however, it cannot in general be computed from the public pair (ek, c) alone, since any public procedure extracting the message with noticeably different behavior on real codewords and on uniform strings immediately violates pseudorandomness.

Lemma 4 (Public Extractability Breaks Pseudorandomness). *Let $\mathsf{PRC} = (\mathtt{KGen}, \mathtt{Encode}, \mathtt{Decode})$ be a public-key pseudorandom code. Suppose there are PPT algorithms $\mathcal{A}$ and $\mathtt{Ext}$ such that,*

$$\left| \Pr\left[\mathtt{Ext}(\mathsf{ek}, c) = m \;\middle|\; \begin{array}{l} (\mathsf{ek}, \mathsf{dk}) \leftarrow \mathtt{KGen}(1^\lambda), \\ m \leftarrow \mathcal{A}(1^\lambda, \mathsf{ek}), \\ c \leftarrow \mathtt{Encode}(\mathsf{ek}, m) \end{array} \right] - \Pr\left[\mathtt{Ext}(\mathsf{ek}, c) = m \;\middle|\; \begin{array}{l} (\mathsf{ek}, \mathsf{dk}) \leftarrow \mathtt{KGen}(1^\lambda), \\ m \leftarrow \mathcal{A}(1^\lambda, \mathsf{ek}), \\ c \leftarrow\!\!\$\ \Sigma^\star \end{array} \right] \right|$$

is non-negligible. Then PRC is not pseudorandom.

Proof. A pseudorandomness distinguisher chooses $m \leftarrow \mathcal{A}(1^\lambda, \mathsf{ek})$, queries its challenge oracle on m to obtain c, and outputs 1 iff $\mathtt{Ext}(\mathsf{ek}, c) = m$. The displayed gap is exactly its distinguishing advantage. □

Thus a public-key UC realization must *unavoidably* equip $\mathcal{S}_{\mathsf{PRC}}$ with some non-public explanation source, for example, the hidden decoder state dk as a trapdoor, as in the proof of Theorem 2, or an idealized model such as a RO transcript, as in the proof of CCA security in [AAC+25].

Proposition 7 (Public-Key Explanation Dichotomy). *Any UC realization of the public-key variant of $\mathcal{F}_{\mathsf{PRC}}$ must either* (i) *provide $\mathcal{S}_{\mathsf{PRC}}$ with a trapdoor for ek, or* (ii) *rely on an idealized setup that supplies the explanation. In particular, there is no plain-model realization in which $\mathcal{S}_{\mathsf{PRC}}$ explains accepted fresh codewords from (ek, c) alone.*

Proof. A fresh valid c exists with no corruption, since anyone holding public ek encodes. By Lemma 3, a realizing $\mathcal{S}_{\mathsf{PRC}}$ must return the decoded m. Computing m from (ek, c) alone would make $\mathcal{S}_{\mathsf{PRC}}$ a public extractor $\mathtt{Ext}(\mathsf{ek}, c) = m$, contradicting Lemma 4; hence its answer depends on a trapdoor for ek or on a setup-supplied state. □

This dichotomy is illustrated by recent standard-model CCA-secure public-key PRCs [DJK+26, CMY26] which use a trapdoor instead of a random oracle. At a high level, both replace the FO-based transform of [AAC+25] with the Koppula–Waters template [KW19], where decryption recovers the hinting-PRG seed from secret keys.

PRC-Based Watermark is Not Publicly Detectable. The conclusive message of this section is that a PRC, unlike a PKE, cannot be made *publicly checkable*. A formally analogous dichotomy of Proposition 7 holds for UC-secure public-key encryption, whose ideal-process adversary likewise samples $(\mathsf{pk}, \mathsf{sk})$ itself and decrypts fresh ciphertexts with sk, or relies on an idealized setup like a random oracle; there, however, the statement is uninteresting, since that plaintexts cannot be recovered from $(\mathsf{pk}, \mathsf{ct})$ alone is the design goal of encryption. For a PRC the same statement has bite: one might hope for publicly checkable decoding under a public ek, which Proposition 7 rules out.

The contrast is sharpest for zero-knowledge techniques. Attaching a publicly verifiable NIZK to a ciphertext, as in the Naor–Yung paradigm [NY90], is a standard route to CCA security, because encryption only hides *which* plaintext a ciphertext carries, so a public proof about a ciphertext reveals nothing that must stay hidden. A PRC instead requires that codewords be indistinguishable from uniform strings, so *any* publicly evaluable verdict, including a NIZK verifier for a proof that a string decodes under a committed dk, is itself an extractor in the sense of Lemma 4; indeed, even the mere presence of an accompanying proof already marks a string as watermarked.

Nor does an idealized setup generically restore public verifiability: the setup branch of Proposition 7 supplies the explanation to the *ideal-process adversary* alone, whereas public verification would make the verdict evaluable by every party, and hence every distinguisher, so the extractor argument of Lemma 4 applies unchanged whenever verifiers and distinguishers access the setup symmetrically, such as with a random oracle. A trusted resource can offer only the weaker *verdict delegation*, where the trapdoor holder publishes or proves in zero knowledge the verdicts of individual strings, leaving unadjudicated codewords pseudorandom.

In sum, detection of PRC-based watermarks is an inherently *designated-verifier* functionality, mediated by the trapdoor holder or an idealized setup, and a *publicly* detectable watermark must depart from pseudorandomness altogether.

Acknowledgment. Ganyuan Cao is supported by the ANR-23-CE39-0009 TRUST project. We thank Sébastien Canard, Qingju Wang, Duong Hieu Phan, San Ling, and Dinh Duy Nguyen for their valuable discussions. We thank the anonymous reviewers of CANS 2026 for their careful reading and constructive feedback, which helped improve this work.

References

AAC+25. Omar Alrabiah, Prabhanjan Ananth, Miranda Christ, Yevgeniy Dodis, and Sam Gunn. Ideal pseudorandom codes. In Michal Koucký and Nikhil Bansal, editors, *57th ACM STOC*, pages 1638–1647. ACM Press, June 2025. doi:10.1145/3717823.3718309.

BFKL94. Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 278–291. Springer, Berlin, Heidelberg, August 1994. doi:10.1007/3-540-48329-2_24.

BGI+01. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Berlin, Heidelberg, August 2001. doi:10.1007/3-540-44647-8_1.

BW13. Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Berlin, Heidelberg, December 2013. doi:10.1007/978-3-642-42045-0_15.

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001. doi:10.1109/SFCS.2001.959888.

CDG+18. Jan Camenisch, Manu Drijvers, Tommaso Gagliardoni, Anja Lehmann, and Gregory Neven. The wonderful world of global random oracles. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 280–312. Springer, Cham, April / May 2018. doi:10.1007/978-3-319-78381-9_11.

CFGN96. Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *28th ACM STOC*, pages 639–648. ACM Press, May 1996. doi:10.1145/237814.238015.

CG24. Miranda Christ and Sam Gunn. Pseudorandom error-correcting codes. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VI*, volume 14925 of *LNCS*, pages 325–347. Springer, Cham, August 2024. doi:10.1007/978-3-031-68391-6_10.

CGG+25. Miranda Christ, Noah Golowich, Sam Gunn, Ankur Moitra, and Daniel Wichs. Improved pseudorandom codes from permuted puzzles. Cryptology ePrint Archive, Report 2025/2222, 2025. URL: https://eprint.iacr.org/2025/2222.

CHK05. Ran Canetti, Shai Halevi, and Jonathan Katz. Adaptively-secure, non-interactive public-key encryption. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 150–168. Springer, Berlin, Heidelberg, February 2005. doi:10.1007/978-3-540-30576-7_9.

CHS25. Aloni Cohen, Alexander Hoover, and Gabe Schoenbach. Watermarking language models for many adaptive users. In Marina Blanton, William Enck, and Cristina Nita-Rotaru, editors, *2025 IEEE Symposium on Security and Privacy*, pages 2583–2601. IEEE Computer Society Press, May 2025. doi:10.1109/SP61157.2025.00084.

CLNS17. Jan Camenisch, Anja Lehmann, Gregory Neven, and Kai Samelin. UC-secure non-interactive public-key encryption. In Boris Köpf and Steve Chong, editors, *CSF 2017 Computer Security Foundations Symposium*, pages 217–233. IEEE Computer Society Press, 2017. doi:10.1109/CSF.2017.14.

CMY26. Yu Chen, Xinyu Mao, and Hongxu Yi. Advancing pseudorandom codes: Beyond parity checks and standard-model CCA1 security. Cryptology ePrint Archive, Paper 2026/1228, 2026. URL: https://eprint.iacr.org/2026/1228.

CS82. John H. Conway and Neil J. A. Sloane. Fast quantizing and decoding algorithms for lattice quantizers and codes. *IEEE Transactions on Information Theory*, 28(2):227–232, 1982.

CS02. Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Berlin, Heidelberg, April / May 2002. doi:10.1007/3-540-46035-7_4.

DFV26. Victor Dyseryn, Danilo Francati, and Daniele Venturi. Public-key pseudorandom codes from distorted mceliece assumptions. Cryptology ePrint Archive, Paper 2026/1184, 2026. URL: https://eprint.iacr.org/2026/1184.

DJK+26. Nico Döttling, Antoine Joux, Venkata Koppula, Mahesh Sreekumar Rajasree, and Hendrik Waldner. Chosen ciphertext secure pseudorandom codes in the standard model. In *CRYPTO*, 2026. URL: https://eprint.iacr.org/2026/1227.

DKRR26. Nico Döttling, Venkata Koppula, Mahesh Sreekumar Rajasree, and Felix Rohrbach. On robust combiners for pseudorandom codes and PKE with pseudorandom ciphertexts. TCC, 2026. URL: https://eprint.iacr.org/2026/2650.

DMR25. Nico Döttling, Anne Müller, and Mahesh Sreekumar Rajasree. Separating pseudorandom codes from local oracles. In Benny Applebaum and Huijia (Rachel) Lin, editors, *TCC 2025, Part IV*, volume 16271 of *LNCS*, pages 225–257. Springer, Cham, December 2025. doi:10.1007/978-3-032-12290-2_8.

DN00. Ivan Damgård and Jesper Buus Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 432–450. Springer, Berlin, Heidelberg, August 2000. doi:10.1007/3-540-44598-6_27.

FGP+26. Danilo Francati, Yevin Nikhel Goonatilake, Shubham Pawar, Daniele Venturi, and Giuseppe Ateniese. The coding limits of robust watermarking for generative models. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2026. URL: https://eprint.iacr.org/2025/1620.

FO99. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554. Springer, Berlin, Heidelberg, August 1999. doi:10.1007/3-540-48405-1_34.

GG24. Surendra Ghentiyala and Venkatesan Guruswami. New constructions of pseudorandom codes. Cryptology ePrint Archive, Report 2024/1425, 2024. URL: https://eprint.iacr.org/2024/1425.

GGW25. Sanjam Garg, Sam Gunn, and Mingyuan Wang. Black-box crypto is useless for pseudorandom codes. In Benny Applebaum and Huijia (Rachel) Lin, editors, *TCC 2025, Part IV*, volume 16271 of *LNCS*, pages 205–224. Springer, Cham, December 2025. doi:10.1007/978-3-032-12290-2_7.

GM24. Noah Golowich and Ankur Moitra. Edit distance robust watermarks via indexing pseudorandom codes. *Advances in Neural Information Processing Systems*, 37:20645–20693, 2024.

GZS24. Sam Gunn, Xuandong Zhao, and Dawn Song. An undetectable watermark for generative image models. Cryptology ePrint Archive, Report 2024/1597, 2024. URL: https://eprint.iacr.org/2024/1597.

JL00. Stanislaw Jarecki and Anna Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 221–242. Springer, Berlin, Heidelberg, May 2000. doi:10.1007/3-540-45539-6_16.

KW19. Venkata Koppula and Brent Waters. Realizing chosen ciphertext security generically in attribute-based encryption and predicate encryption. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 671–700. Springer, Cham, August 2019. doi:10.1007/978-3-030-26951-7_23.

LSS26. Huijia Lin, Kameron Shahabi, and Min Jae Song. Unforgeable watermarks for language models via robust signatures. In *CRYPTO*, 2026. URL: https://eprint.iacr.org/2026/282.

Nie02. Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 111–126. Springer, Berlin, Heidelberg, August 2002. doi:10.1007/3-540-45708-9_8.

NY90. Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990. doi:10.1145/100216.100273.

RS60. Irving S Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.

SW14. Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014. doi:10.1145/2591796.2591825.

Vit67. Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

A Background on Ideal Pseudorandom Code

We recall the construction pipeline of [CG24, AAC+25] only to the extent needed for our adaptive-corruption separation. The construction begins with an LDPC-based zero-bit PRC (Construction 5), converts it to a single-bit (Construction 6) and then multi-bit PRC in public-key setting (Construction 8) and in secret-key setting (Construction 7), and finally applies an FO-style transform to obtain ideal security in the secret-key setting and CCA security in the public-key setting (Fig. 7).

Definition 15 (Low-Density Parity-Check (LDPC) Code [CG24]). *For integers n, t, r, d, define*

$$\mathcal{S}_{t,n} = \{s \in \mathbb{F}_2^n : \mathsf{wt}(s) = t\}$$

as the set of all t-sparse vectors in $\mathbb{F}_2^n$, and

$$\mathcal{S}_{t,r,n} = \{H \in \mathbb{F}_2^{r \times n} : \mathsf{wt}(H_{i,:}) = t,\ \forall i \in [r]\}.$$

The distribution $\mathsf{LDPC}[n, d, t, r]$ over $(H, G) \in \mathbb{F}_2^{r \times n} \times \mathbb{F}_2^{n \times d}$ is defined as:

1. *Sample $H \leftarrow\!\!\$\ \mathcal{S}_{t,r,n}$, i.e., $H \in \mathbb{F}_2^{r \times n}$ is chosen to have i.i.d. random t-sparse rows.*
2. *Sample $G \leftarrow\!\!\$\ (\ker H)^d$, i.e., $G \in \mathbb{F}_2^{n \times d}$ is a random matrix subject to $HG = 0$.*
3. *Output (H, G).*

Construction 5 (Zero-Bit LDPC-Based PRC [CG24])**.** *Let n, d, t, r be parameter functions and let $\eta, \zeta \in [0, \frac{1}{2})$ be noise and decoding-threshold parameters. The zero-bit LDPC-based PRC $\mathsf{LDPC\text{-}PRC}_0[n, d, t, r, \eta, \zeta]$ has message space $\{1\}$ and is defined as follows.*

- $\mathtt{KGen}(1^\lambda)$*: Sample $(H, G) \leftarrow\!\!\$\ \mathsf{LDPC}[n, d, t, r]$ and $z \leftarrow\!\!\$\ \mathbb{F}_2^n$, output $\mathsf{dk} := (H, z)$ and $\mathsf{ek} := (G, z)$.*
- $\mathtt{Encode}(\mathsf{ek}, 1)$*: Parse $\mathsf{ek} = (G, z)$. Sample $u \leftarrow\!\!\$\ \mathbb{F}_2^d$ and $e \leftarrow\!\!\$\ \mathcal{S}_{\eta n,n}$. Output $c := Gu \oplus e \oplus z$.*
- $\mathtt{Decode}(\mathsf{dk}, x)$*: Parse $\mathsf{dk} = (H, z)$. Compute $s := H(x \oplus z)$. If $\mathsf{wt}(s) < (\frac{1}{2} - \zeta) \cdot r$, output 1; otherwise output $\perp$.*

Construction 6 (Single-Bit LDPC-Based PRC [CG24])**.** *Let $\mathsf{LDPC\text{-}PRC}_0[n, d, t, r, \eta, \zeta]$ be the zero-bit LDPC-based PRC from Construction 5. The single-bit LDPC-based PRC $\mathsf{LDPC\text{-}PRC}_1[n, d, t, r, \eta, \zeta]$ has message space $\{0, 1\}$ and is defined as follows.*

- $\mathtt{KGen}(1^\lambda)$*: For each $b \in \{0, 1\}$, sample $(\mathsf{dk}_b, \mathsf{ek}_b) \leftarrow \mathsf{LDPC\text{-}PRC}_0.\mathtt{KGen}(1^\lambda)$, where $\mathsf{dk}_b = (H_b, z_b)$ and $\mathsf{ek}_b = (G_b, z_b)$. Output $\mathsf{dk} := (\mathsf{dk}_0, \mathsf{dk}_1)$ and $\mathsf{ek} := (\mathsf{ek}_0, \mathsf{ek}_1)$.*
- $\mathtt{Encode}(\mathsf{ek}, m)$*: Output $c \leftarrow \mathsf{LDPC\text{-}PRC}_0.\mathtt{Encode}(\mathsf{ek}_m, 1)$ for $\mathsf{ek} = (\mathsf{ek}_0, \mathsf{ek}_1)$.*
- $\mathtt{Decode}(\mathsf{dk}, x)$*: Parse $\mathsf{dk} = (\mathsf{dk}_0, \mathsf{dk}_1)$ and let $b_0 := \mathsf{LDPC\text{-}PRC}_0.\mathtt{Decode}(\mathsf{dk}_0, x)$ and $b_1 := \mathsf{LDPC\text{-}PRC}_0.\mathtt{Decode}(\mathsf{dk}_1, x)$. If $b_0 = 1$ and $b_1 = \perp$, output 0. If $b_1 = 1$ and $b_0 = \perp$, output 1. Otherwise, output $\perp$.*

Construction 7 (Secret-Key Multi-Bit PRC [AAC+25])**.** *Let $\mathsf{PRC}_1 = (\mathtt{KGen}, \mathtt{Encode}, \mathtt{Decode})$ be a secret-key single-bit PRC with codeword length n. Let $\mathsf{ECC} = (\mathtt{Encode}, \mathtt{Decode})$ be an error-correcting code, let $\mathsf{PRG} : \{0, 1\}^\ell \to \{0, 1\}^\omega$ be a pseudorandom generator, and let π be a permutation over the final codeword coordinates. The secret-key multi-bit PRC $\mathsf{PRC}_{\mathsf{mb}}^{\mathsf{sk}}$ is defined as follows.*

- $\mathtt{KGen}(1^\lambda)$*: Generate $\mathsf{sk}_1 \leftarrow \mathsf{PRC}_1.\mathtt{KGen}(1^\lambda)$ and sample a permutation π. Output $\mathsf{ek} := \mathsf{dk} := (\mathsf{sk}_1, \pi)$.*

– $\mathtt{Encode}(\mathsf{ek}, m)$: *Parse* $\mathsf{ek} = (\mathsf{sk}_1, \pi)$. *Sample* $r = (r_1, \ldots, r_\ell) \leftarrow\!\!\$\ \{0,1\}^\ell$. *For every* $j \in [\ell]$, *compute* $c_j \leftarrow \mathsf{PRC}_1.\mathtt{Encode}(\mathsf{sk}_1, r_j)$. *Compute* $w := \mathsf{ECC}.\mathtt{Enc}(m) \oplus \mathsf{PRG}(r)$. *Output* $c := \pi(c_1 \| \cdots \| c_\ell \| w)$.
– $\mathtt{Decode}(\mathsf{dk}, x)$: *Parse* $\mathsf{dk} = (\mathsf{sk}_1, \pi)$ *and compute* $x' := \pi^{-1}(x)$. *Parse* $x' = (c_1 || \ldots || c_\ell || w)$. *For each* $j \in [\ell]$, *compute* $r'_j := \mathsf{PRC}_1.\mathtt{Decode}(\mathsf{sk}_1, c_j)$. *Recover* r' *from* $(r'_1, \ldots, r'_\ell)$, *compute* $M' := w \oplus \mathsf{PRG}(r')$, *and output* $m' := \mathsf{ECC}.\mathtt{Dec}(M')$. *If any decoding step fails, output* $\perp$.

Construction 8 (Public-Key Multi-Bit PRC [AAC+25])**.** *Let* $\mathsf{PRC}_1 = (\mathtt{KGen}, \mathtt{Encode}, \mathtt{Decode})$ *be a public-key single-bit PRC with codeword length* n. *Let* ECC_1 *and* ECC_2 *be error-correcting codes, and let* $\mathsf{PRG} : \{0,1\}^\kappa \to \{0,1\}^\omega$ *be a pseudorandom generator. The public-key multi-bit PRC* $\mathsf{PRC}^{\mathsf{pk}}_{\mathsf{mb}}$ *is defined as follows.*

– $\mathtt{KGen}(1^\lambda)$: *Generate* $(\mathsf{ek}_1, \mathsf{dk}_1) \leftarrow \mathsf{PRC}_1.\mathtt{KGen}(1^\lambda)$. *Output* $\mathsf{ek} := \mathsf{ek}_1$ *and* $\mathsf{dk} := \mathsf{dk}_1$.
– $\mathtt{Encode}(\mathsf{ek}, m)$: *Sample* $r \leftarrow\!\!\$\ \{0,1\}^\kappa$ *and compute* $a := \mathsf{ECC}_1.\mathtt{Encode}(r) \in \{0,1\}^\ell$. *For every* $j \in [\ell]$, *compute* $c_j \leftarrow \mathsf{PRC}_1.\mathtt{Encode}(\mathsf{ek}, a_j)$. *Compute* $w \leftarrow \mathsf{ECC}_2.\mathtt{Encode}(m) \oplus \mathsf{PRG}(r)$. *Output* $c := c_1 || \ldots || c_\ell || w$.
– $\mathtt{Decode}(\mathsf{dk}, c)$: *Parse* $c = c_1 || \ldots || c_\ell || w$. *For every* $j \in [\ell]$, *compute* $a'_j := \mathsf{PRC}_1.\mathtt{Decode}(\mathsf{dk}, c_j)$. *Compute* $r' := \mathsf{ECC}_1.\mathtt{Dec}(a'_1, \ldots, a'_\ell)$. *If this fails, output* $\perp$. *Compute* $M' := w \oplus \mathsf{PRG}(r')$ *and output* $m' \leftarrow \mathsf{ECC}_2.\mathtt{Dec}(M')$. *If this final decoding step fails, output* $\perp$.

Transform to Ideal/CCA Security. In Fig. 7, we recall the Fujisaki–Okamoto (FO) [FO99]-style transform to ideal/CCA security of [AAC+25]. Whereas the original secret-key transform derives R via a secret pseudorandom function $\mathsf{PRF}_K(r||m)$, we instead use the global random oracle $\mathcal{G}_{\mathsf{RO}}$, keeping the secret-key and public-key variants syntactically consistent within the same $\mathcal{G}_{\mathsf{RO}}$-hybrid model for simplicity. This does not affect our conclusion that Π_{PRC} fails to realize $\mathcal{F}_{\mathsf{PRC}}$ under adaptive corruption, since the LDPC code itself has low opening capacity and thus admits no decoder-non-committing pseudorandom code.

We use the *restricted programmable and observable random oracle* (rpoRO) of Camenisch et al. [CDG+18] (Fig. 8). It can be queried by a party instance $(\mathcal{P}_{\mathsf{pid}}, \mathsf{sid})$ or by the adversary, and is *domain-separated*: every query has the form (sid, x), with out-of-session or direct adversarial queries deemed *illegitimate*, so that $\mathcal{S}_{\mathsf{PRC}}$ can observe the queries $\mathcal{A}$ makes in other sessions for simulation. $\mathcal{G}_{\mathsf{RO}}$ also offers a programming interface, included only for completeness since our proofs do not use it.

B Building Blocks for Non-Committing PRCs

We recall the primitives underlying the constructions in Section 6.

Definition 16 (Puncturable PRF). *A* puncturable pseudorandom function $\mathsf{pPRF} = (\mathtt{Setup}, \mathtt{Puncture}, \mathtt{Eval}, \mathtt{PuncEval})$ *with domain* $\mathcal{X}$ *and range* $\mathcal{Y}$ *is a tuple of PPT algorithms:*

Protocol: Π_{PRC}

Participants: an encoder $\mathcal{P}_{\mathsf{ecd}}$; a decoder $\mathcal{P}_{\mathsf{dcd}}$.

– $\langle \texttt{KeyGen} : \mathsf{sid} \rangle$ at $\mathcal{P}_{\mathsf{dcd}}$ or $\mathcal{P}_{\mathsf{ecd}}$:
1. Compute $(\overline{\mathsf{ek}}, \overline{\mathsf{dk}}) \leftarrow \overline{\mathsf{PRC}}.\texttt{KGen}(1^\lambda)$.
2. Output $\{\texttt{"key"} : \mathsf{sid}, \overline{\mathsf{ek}}\}$ to $\mathcal{P}_{\mathsf{ecd}}$ and $\mathcal{P}_{\mathsf{dcd}}$.
3. Store $(\mathsf{sid}, \overline{\mathsf{ek}})$ locally.
4. Store $(\mathsf{sid}, \overline{\mathsf{dk}})$ locally.

– $\langle \texttt{Encode} : \mathsf{ecd}, \mathsf{sid}, \overline{\mathsf{ek}}, m \rangle$ at $\mathcal{P}_{\mathsf{ecd}}$:
1. Sample $r \leftarrow\!\$\ \{0,1\}^\lambda$
2. Query $\mathcal{G}_{\mathsf{RO}}$ in sid with $r||m$ to get $R = R_1||R_2$.
3. Query $\mathcal{G}_{\mathsf{RO}}$ in sid to check if any $(r||m, R)$ is maliciously programmed by the adversary. If yes, abort.
4. Compute $c \leftarrow \overline{\mathsf{PRC}}.\texttt{Encode}(\overline{\mathsf{ek}}, r||m||R_2; R_1)$
5. Output $\{\texttt{"code"} : \mathsf{ecd}, \mathsf{sid}, c\}$ to $\mathcal{P}_{\mathsf{ecd}}$.

– $\langle \texttt{Decode} : \mathsf{dcd}, \mathsf{sid}, c \rangle$ at $\mathcal{P}_{\mathsf{dcd}}$:
1. Compute $y \leftarrow \overline{\mathsf{PRC}}.\texttt{Decode}(\overline{\mathsf{dk}}, c)$ with locally stored dk.
2. If $y = \perp$, abort. Otherwise, parse $r||m||R_2 := y$.
3. Query $\mathcal{G}_{\mathsf{RO}}$ in sid with $r||m$ to get $R = R_1||R_2'$.
4. Query $\mathcal{G}_{\mathsf{RO}}$ in sid to check if any $(r||m, R)$ is maliciously programmed by the adversary. If yes, abort.
5. Compute $c' \leftarrow \overline{\mathsf{PRC}}.\texttt{Encode}(\overline{\mathsf{ek}}, r||m||R_2; R_1)$.
6. If $R_2 = R_2' \wedge \mathsf{dist}(c, c') < \delta$, output $\{\texttt{"word"} : \mathsf{dcd}, \mathsf{sid}, c, m\}$.
7. Otherwise, abort.

Fig. 7: Protocol Π_{PRC} in presence of a global $\mathcal{G}_{\mathsf{RO}}$ adapted from [AAC+25] instantiated with the multi-bit pseudorandom code $\overline{\mathsf{PRC}}$ from [CG24]. The highlight part appears exclusively when PRC is public-key i.e., ek is public. The framed part appears exclusively when PRC is secret-key. The secret-key is adapted from [CG24] by replacing PRF with $\mathcal{G}_{\mathsf{RO}}$.

Functionality: $\mathcal{G}_{\mathsf{RO}}$

Parameter: output length λ.
Participants: a party instance $(\mathcal{P}_{\mathsf{pid}}, \mathsf{sid})$ and an adversary.

– $\langle \texttt{Query} : x \rangle$ from $(\mathcal{P}_{\mathsf{pid}}, \mathsf{sid})$ or the adversary:
1. If there does not exist v such that (x, v) is in the query list Qry.
   a. Sample v from $\{0,1\}^\lambda$.
   b. Append (x, v) to Qry.
2. Parse x as (sid', x').
3. If $\mathsf{sid} \neq \mathsf{sid}'$ or the query is from the adversary, append $((\mathsf{sid}', x'), v)$ to illegitimate query list $\mathsf{Illegal}$ under the label sid'.
4. Output $\{\texttt{"val"} : v\}$ to the caller.

– $\langle \texttt{Observe} : \mathsf{sid} \rangle$ from the adversary:
1. Get the illegal trace tr from $\mathsf{Illegal}$ under the label sid.
2. Output $\{\texttt{"obv"} : \mathsf{sid}, \mathsf{tr}\}$ to the adversary.

– $\langle \texttt{ProgRO} : x, v \rangle$ from the adversary:
1. If there does not exist v' such that (x, v') in Qry and $v' \neq v$:
   a. Append (x, v) to Qry.
   b. Append x to the list of programmed queries Prog.
   c. Output $\{\texttt{"cfm"} : 1\}$ to the adversary.
2. Otherwise, output $\{\texttt{"cfm"} : 0\}$ to the adversary.

– $\langle \texttt{IsProg} : x \rangle$ from $(\mathcal{P}_{\mathsf{pid}}, \mathsf{sid})$ or the adversary:
1. Parse x as (sid', x').
2. If it is from $(\mathcal{P}_{\mathsf{pid}}, \mathsf{sid})$ and $\mathsf{sid} \neq \mathsf{sid}'$, abort.
3. Set $b := x \in_? \mathsf{Prog}$.
4. Output $\{\texttt{"chk"} : b\}$ to $(\mathcal{P}_{\mathsf{pid}}, \mathsf{sid})$ or the adversary.

Fig. 8: A restricted programmable and observable global random oracle (rpoRO) $\mathcal{G}_{\mathsf{RO}}$ described by Camenisch et al. [CDG+18].

– $\mathsf{msk} \leftarrow \mathtt{Setup}(1^\lambda)$*: on input a security parameter* λ*, outputs a master secret key* msk*.*
– $\mathsf{psk} \leftarrow \mathtt{Puncture}(\mathsf{msk}, S)$*: on input a set* $S \subseteq \mathcal{X}$*, outputs a punctured key* psk*.*
– $y \leftarrow \mathtt{Eval}(\mathsf{msk}, x)$*: on input* $x \in \mathcal{X}$*, outputs* $y \in \mathcal{Y}$*.*
– $y \leftarrow \mathtt{PuncEval}(\mathsf{psk}, x)$*: on input* $x \in \mathcal{X} \setminus S$*, outputs* $y \in \mathcal{Y}$*.*

We require pPRF *to satisfy the following two properties.*

– (Correctness): *For every* $S \subseteq \mathcal{X}$*, every* $\mathsf{msk} \leftarrow \mathsf{pPRF}.\mathtt{Setup}(1^\lambda)$*, every* $\mathsf{psk} \leftarrow \mathsf{pPRF}.\mathtt{Puncture}(\mathsf{msk}, S)$*, and every* $x \in \mathcal{X} \setminus S$*,*

$$\Pr[\mathsf{pPRF}.\mathtt{PuncEval}(\mathsf{psk}, x) \neq \mathsf{pPRF}.\mathtt{Eval}(\mathsf{msk}, x)] \leq \mathsf{negl}(\lambda).$$

– (Pseudorandomness at Punctured Points): *For every PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*,*

$$\left|\Pr\left[\mathcal{A}_2(\mathsf{st}, \mathsf{psk}, \{y_{0,i}\}_{i\in[q]}) = 1\right] - \Pr\left[\mathcal{A}_2(\mathsf{st}, \mathsf{psk}, \{y_{1,i}\}_{i\in[q]}) = 1\right]\right| \leq \mathsf{negl}(\lambda),$$

taken over $(S = \{x_1, \ldots, x_q\}, \mathsf{st}) \leftarrow \mathcal{A}_1(1^\lambda)$ *with distinct* $x_i \in \mathcal{X}$*,* $\mathsf{msk} \leftarrow \mathsf{pPRF}.\mathtt{Setup}(1^\lambda)$*,* $\mathsf{psk} \leftarrow \mathsf{pPRF}.\mathtt{Puncture}(\mathsf{msk}, S)$*,* $y_{0,i} \leftarrow \mathsf{pPRF}.\mathtt{Eval}(\mathsf{msk}, x_i)$ *for* $i \in [q]$*, and* $y_{1,1}, \ldots, y_{1,q} \leftarrow\!\!\$\ \mathcal{Y}$ *are uniform and independent.*

Definition 17 (Indistinguishability Obfuscator). *A PPT algorithm* iO *is an* indistinguishability obfuscator *for a circuit class* $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ *if the following hold.*

– (Correctness.) *For every* $\lambda \in \mathbb{N}$*, every* $C \in \mathcal{C}_\lambda$*, and every input* x*,*

$$\Pr\left[C'(x) = C(x) \ : \ C' \leftarrow \mathsf{iO}(C)\right] = 1.$$

– (Indistinguishability.) *For every PPT adversary* $\mathcal{A}$ *and every pair* $C_0, C_1 \in \mathcal{C}_\lambda$ *such that* $C_0(x) = C_1(x)$ *for all inputs* x *and* $|C_0| = |C_1|$*,*

$$\left|\Pr\left[\mathcal{A}\big(\mathsf{iO}(C_0)\big) = 1\right] - \Pr\left[\mathcal{A}\big(\mathsf{iO}(C_1)\big) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Definition 18 (Smooth Projective Hash Family). *Let* $L \subseteq U$ *be sets, and let* $\mathcal{R} \subseteq U \times W$ *be a relation for a witness set* W *such that* $L = \{u \in U : \exists\, w \in W,\ (u, w) \in \mathcal{R}\}$*. A* projective hash family $\mathsf{SPHF} = (\mathtt{HashKG}, \mathtt{ProjKG}, \mathtt{Hash}, \mathtt{Proj})$ *over* (U, L) *with hash space* $\mathcal{H}$ *is a tuple of PPT algorithms:*

– $\mathsf{hk} \leftarrow \mathtt{HashKG}(1^\lambda)$*: on input a security parameter* λ*, outputs a hashing key* hk*.*
– $\mathsf{hp} \leftarrow \mathtt{ProjKG}(\mathsf{hk})$*: on input a hashing key* hk*, outputs a projection key* hp*.*
– $h \leftarrow \mathtt{Hash}(\mathsf{hk}, u)$*: on input* $u \in U$*, outputs* $h \in \mathcal{H}$*.*
– $h \leftarrow \mathtt{Proj}(\mathsf{hp}, u, w)$*: on input* $u \in L$ *and a witness* $w \in W$ *with* $(u, w) \in \mathcal{R}$*, outputs* $h \in \mathcal{H}$*.*

We require SPHF *to satisfy the following properties.*

– (Projectivity.) *For all* $\mathsf{hk} \leftarrow \mathsf{SPHF}.\mathtt{HashKG}(1^\lambda)$*,* $\mathsf{hp} \leftarrow \mathsf{SPHF}.\mathtt{ProjKG}(\mathsf{hk})$*, and* $(u, w) \in \mathcal{R}$*,*

$$\Pr[\mathsf{SPHF}.\mathtt{Hash}(\mathsf{hk}, u) \neq \mathsf{SPHF}.\mathtt{Proj}(\mathsf{hp}, u, w)] \leq \mathsf{negl}(\lambda).$$

– (Sparsity.)
$$\Pr_{u \leftarrow\$ U}[u \in L] \leq \mathsf{negl}(\lambda).$$
– (Subset Membership Indistinguishability.) *For every PPT adversary* $\mathcal{A}$,
$$\left| \Pr_{(u,w) \leftarrow\$ \mathcal{R}}\left[\mathcal{A}(1^\lambda, u) = 1\right] - \Pr_{u \leftarrow\$ U}\left[\mathcal{A}(1^\lambda, u) = 1\right] \right| \leq \mathsf{negl}(\lambda).$$
where $(u, w) \leftarrow\$ \mathcal{R}$ *is a PPT algorithm sampling pairs* $(u, w) \in U \times W$ *such that* $(u, w) \in \mathcal{R}$.
– (q-Smoothness.) *For all distinct* $u_1, \ldots, u_q \in U \setminus L$,
$$(\mathsf{hp},\ \mathsf{SPHF.Hash}(\mathsf{hk}, u_1), \ldots, \mathsf{SPHF.Hash}(\mathsf{hk}, u_q)) \quad \textit{and} \quad (\mathsf{hp},\ h_1, \ldots, h_q)$$
are statistically close, where $\mathsf{hk} \leftarrow \mathsf{SPHF.HashKG}(1^\lambda)$, $\mathsf{hp} \leftarrow \mathsf{SPHF.ProjKG}(\mathsf{hk})$, *and* $h_1, \ldots, h_q \leftarrow\$ \mathcal{H}$ *are sampled uniformly and independently.*

C Full Proofs of Theorems

C.1 Proof of Theorem 1

Theorem 1. *Let* $\mathsf{PRC} = (\mathtt{KGen}, \mathtt{Encode}, \mathtt{Decode})$ *be a secret-key pseudorandom code with distance bound* δ, *and let* Π_{PRC} *be the natural protocol in which* $\mathcal{P}_{\mathsf{ecd}}$ *runs* $\mathsf{PRC.Encode}(\mathsf{ek}, \cdot)$ *and* $\mathcal{P}_{\mathsf{dcd}}$ *runs* $\mathsf{PRC.Decode}(\mathsf{dk}, \cdot)$ *using honestly generated secret state* $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PRC.KGen}(1^\lambda)$. *Then* Π_{prc} *realizes* $\mathcal{F}_{\mathsf{PRC}}$ *against an adversary that non-adaptively corrupts parties, if and only if* PRC *satisfies the secret-key ideal security as in Fig. 1, up to the standard translation between UC environments and oracle-game adversaries.*

Proof. We prove both directions. If $\mathcal{P}_{\mathsf{dcd}}$ is corrupted, whether or not $\mathcal{P}_{\mathsf{ecd}}$ is, then $\mathcal{F}_{\mathsf{PRC}}$ lets $\mathcal{S}_{\mathsf{PRC}}$ produce every codeword, which it does by running $\mathsf{PRC.Encode}$ under the key it generated, and every decoding is performed by the adversary. Hence $\mathcal{S}_{\mathsf{PRC}}$ simply runs Π_{PRC} and the two executions are identically distributed. It remains to consider an honest $\mathcal{P}_{\mathsf{dcd}}$.

($\Rightarrow$) Contrapositively, suppose an environment $\mathcal{Z}$ distinguishes $\mathbf{REAL}_{\Pi_{\mathsf{PRC}}, \mathcal{A}, \mathcal{Z}}$ from $\mathbf{IDEAL}_{\mathcal{F}_{\mathsf{PRC}}, \mathcal{S}_{\mathsf{PRC}}, \mathcal{Z}}$ for a real-world adversary $\mathcal{A}$ with a fixed, non-adaptive corruption set. We build a standalone distinguisher $\mathcal{A}_{ideal}$ against the secret-key ideal security of PRC that internally runs $\mathcal{Z}$ and $\mathcal{A}$ and simulates the UC execution. Since the corruption set is fixed before the session, the simulation fixes from the outset which activations are answered through $\mathcal{O}_{\mathsf{Encode}}$, $\mathcal{O}_{\mathsf{Decode}}$ in $\mathbf{G}^{\mathrm{Ideal}}_{\mathsf{PRC}}$; corrupted-party behavior is generated internally by $\mathcal{A}$ and forwarded. Thus the standalone oracles serve only honest activations: an activation $\langle \mathtt{Encode} : \mathsf{ecd}, \mathsf{sid}, m \rangle$ of an uncorrupted $\mathcal{P}_{\mathsf{ecd}}$ is answered by $\mathcal{O}_{\mathsf{Encode}}(m)$, and $\langle \mathtt{Decode} : \mathsf{dcd}, \mathsf{sid}, x \rangle$ of an uncorrupted $\mathcal{P}_{\mathsf{dcd}}$ by $\mathcal{O}_{\mathsf{Decode}}(x)$.

If only $\mathcal{P}_{\mathsf{ecd}}$ is corrupted, there are no honest encodings, since $\mathcal{F}_{\mathsf{PRC}}$ lets $\mathcal{S}_{\mathsf{PRC}}$ produce every codeword, so $\mathcal{A}_{ideal}$ needs no challenge oracle and the only activations are decoding queries to the honest $\mathcal{P}_{\mathsf{dcd}}$: $\mathcal{A}$ holds $\mathsf{ek} = \mathsf{dk}$ and may

feed $\mathcal{P}_{\mathsf{dcd}}$ a valid c produced outside $\langle \texttt{Encode} : \cdot \rangle$, i.e. with $\mathsf{Near}^{\delta}_{\mathsf{Issued}[\mathsf{sid}]}(c) = \varnothing$. In **REAL**, $\mathcal{P}_{\mathsf{dcd}}$ outputs $\mathsf{PRC}.\texttt{Decode}(\mathsf{dk}, c)$; since $\mathcal{A}_{ideal}$ also holds $\mathsf{ek} = \mathsf{dk}$, it answers by computing $\mathsf{PRC}.\texttt{Decode}(\mathsf{dk}, c)$ itself. In **IDEAL**, this fresh c is delegated by $\mathcal{F}_{\mathsf{PRC}}$ to $\mathcal{S}_{\mathsf{PRC}}$, which holds the same key and returns the same message. Both worlds output $\mathsf{PRC}.\texttt{Decode}(\mathsf{dk}, c)$, so the case is consistent and adds no advantage: the value is one $\mathcal{A}_{ideal}$ could already compute, since non-adaptive corruption exposes the key before any codeword is sampled.

In the real branch ($b = 0$) of $\mathbf{G}^{\mathrm{Ideal}}_{\mathsf{PRC}}$, the oracles run $\mathsf{PRC}.\texttt{Encode}(\cdot)$ and $\mathsf{PRC}.\texttt{Decode}(\cdot)$, so $\mathcal{Z}$'s view is distributed as $\mathbf{REAL}_{\Pi_{\mathsf{PRC}},\mathcal{A},\mathcal{Z}}$. In the ideal branch ($b = 1$), encoding queries return fresh uniform strings recorded in $\mathcal{Q}$ and decoding uses the table-first rule on $\mathcal{Q}$, which is exactly the behavior of $\mathcal{F}_{\mathsf{PRC}}$ on honest activations, so $\mathcal{Z}$'s view is distributed as $\mathbf{IDEAL}_{\mathcal{F}_{\mathsf{PRC}},\mathcal{S}_{\mathsf{PRC}},\mathcal{Z}}$. Hence $\mathcal{A}_{ideal}$ distinguishes the two branches with the same advantage as $\mathcal{Z}$.

($\Leftarrow$) Conversely, let $\mathcal{A}_{ideal}$ break the ideal security of PRC. We build a UC environment $\mathcal{Z}$ that runs $\mathcal{A}_{ideal}$ internally, using the honest UC parties to emulate its oracles: an encoding query m activates $\mathcal{P}_{\mathsf{ecd}}$ on $\langle \texttt{Encode} : \mathsf{ecd}, \mathsf{sid}, m \rangle$, a decoding query x activates $\mathcal{P}_{\mathsf{dcd}}$ on $\langle \texttt{Decode} : \mathsf{dcd}, \mathsf{sid}, x \rangle$, and $\mathcal{Z}$ outputs $\mathcal{A}_{ideal}$'s bit. In the real UC execution the honest parties run Π_{PRC}, so $\mathcal{A}_{ideal}$'s oracle interface matches the real branch ($b = 0$); in the ideal execution honest encodings are uniform strings recorded in $\mathsf{Issued}[\mathsf{sid}]$ and decoding uses the table-first rule, matching the ideal branch ($b = 1$). Thus $\mathcal{Z}$ distinguishes the two UC executions with the same advantage as $\mathcal{A}_{ideal}$.

Hence Π_{PRC} realizes $\mathcal{F}_{\mathsf{PRC}}$ against non-adaptive corruption if and only if PRC satisfies secret-key ideal security in Fig. 1. □

C.2 Proof of Theorem 2

Theorem 2. *Let* $\mathsf{PRC} = (\texttt{KGen}, \texttt{Encode}, \texttt{Decode})$ *be a public-key pseudorandom code with distance bound* δ*, and let* Π_{PRC} *be the natural protocol in which* $\mathcal{P}_{\mathsf{ecd}}$ *runs* $\mathsf{PRC}.\texttt{Encode}(\mathsf{ek}, \cdot)$ *and* $\mathcal{P}_{\mathsf{dcd}}$ *runs* $\mathsf{PRC}.\texttt{Decode}(\mathsf{dk}, \cdot)$ *for honestly generated keys* $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PRC}.\texttt{KGen}(1^{\lambda})$*, with* ek *public. Then* Π_{PRC} *realizes* $\mathcal{F}_{\mathsf{PRC}}$ *against an adversary that non-adaptively corrupts parties, if and only if* PRC *satisfies the* IND-CCA *security as in Fig. 1, up to the standard translation between UC environments and oracle-game adversaries.*

Proof. We use the standard oracle/UC translation, the only extra point being the public-key treatment of fresh decoding queries. As in the proof of Theorem 1, if $\mathcal{P}_{\mathsf{dcd}}$ is corrupted, whether or not $\mathcal{P}_{\mathsf{ecd}}$ is, then $\mathcal{S}_{\mathsf{PRC}}$ produces every codeword by running $\mathsf{PRC}.\texttt{Encode}(\mathsf{ek}, \cdot)$, the adversary performs every decoding, and the two executions coincide, so we consider an honest $\mathcal{P}_{\mathsf{dcd}}$.

For ($\Rightarrow$), argue contrapositively: suppose that for some real-world adversary $\mathcal{A}$ with fixed non-adaptive corruption, an environment $\mathcal{Z}$ distinguishes $\mathbf{REAL}_{\Pi_{\mathsf{PRC}},\mathcal{A},\mathcal{Z}}$ from $\mathbf{IDEAL}_{\mathcal{F}_{\mathsf{PRC}},\mathcal{S}_{\mathsf{PRC}},\mathcal{Z}}$. Build a standalone CCA adversary $\mathcal{A}_{cca}$ that runs $\mathcal{Z}$, $\mathcal{A}$ internally, forwarding the CCA challenger's public key ek to $\mathcal{Z}$. As the corruption pattern is fixed in advance, $\mathcal{A}_{cca}$ knows the honest activations: $\langle \texttt{Encode} : \mathsf{ecd}, \mathsf{sid}, m \rangle$ is answered by $\mathcal{O}_{\mathsf{Encode}}(m)$, $\langle \texttt{Decode} : \mathsf{dcd}, \mathsf{sid}, x \rangle$ by

$\mathcal{O}_{\mathsf{Decode}}(x)$, and corrupted behavior is forwarded unchanged. In the real CCA branch the oracles run $\mathtt{Encode}(\mathsf{ek},\cdot)$, $\mathtt{Decode}(\mathsf{dk},\cdot)$, so $\mathcal{Z}$'s view is $\mathbf{REAL}_{\Pi_{\mathsf{PRC}},\mathcal{A},\mathcal{Z}}$; in the random branch, honest encodings are fresh uniform strings and decoding uses the table-first rule with fresh queries falling back to the real decoder, which is exactly $\mathcal{F}_{\mathsf{PRC}}$ with $\mathcal{S}_{\mathsf{PRC}}$ sampling $(\mathsf{ek},\mathsf{dk}) \leftarrow \mathtt{KGen}(1^\lambda)$, exposing only ek, keeping dk as a trapdoor, and answering delegated fresh x by $\mathtt{Decode}(\mathsf{dk},x)$. Hence $\mathcal{A}_{cca}$ distinguishes the two branches with $\mathcal{Z}$'s advantage.

For ($\Leftarrow$), let $\mathcal{A}_{cca}$ break public-key IND-CCA security. Build a UC environment $\mathcal{Z}$ running $\mathcal{A}_{cca}$ internally: it forwards the public key ek from the simulated key-generation phase and emulates $\mathcal{A}_{cca}$'s oracles by activating the honest encoder on $\langle \mathtt{Encode} : \mathsf{ecd},\mathsf{sid},m\rangle$ and honest decoder on $\langle \mathtt{Decode} : \mathsf{dcd},\mathsf{sid},x\rangle$, outputting $\mathcal{A}_{cca}$'s bit. In the real UC execution $\mathcal{A}_{cca}$ sees the real CCA branch; in the ideal one, honest encodings are uniform strings recorded in $\mathsf{Issued}[\mathsf{sid}]$, decoding uses the table-first rule, and fresh queries are delegated to $\mathcal{S}_{\mathsf{PRC}}$ (which keeps dk and answers by $\mathtt{Decode}(\mathsf{dk},\cdot)$), matching the random branch. Thus $\mathcal{Z}$ distinguishes the two UC executions with $\mathcal{A}_{cca}$'s advantage.

Hence Π_{PRC} realizes $\mathcal{F}_{\mathsf{PRC}}$ against non-adaptive corruptions if and only if PRC satisfies public-key IND-CCA security. □

C.3 Proof of Theorem 3

Theorem 3. *Let* $\mathsf{PRC} = (\mathtt{KGen},\mathtt{Encode},\mathtt{Decode})$ *be a pseudorandom code, and let* Π_{prc} *be its natural protocol as in Theorems 1 and 2. Then* Π_{prc} *realizes* $\mathcal{F}_{\mathsf{PRC}}$ *against adaptive corruptions with decoder-state exposure*[6] *if and only if* PRC *satisfies ideal security (secret-key) or* IND-CCA *security (public-key) and admits a decoder-non-committing pseudorandom code*[7] *compatible with* $\mathcal{F}_{\mathsf{PRC}}$ *that satisfies* IND-OPEN *and, in the secret-key case,* δ*-opening robustness.*

Proof. Throughout, as in the proofs of Theorems 1 and 2, we identify an abort of an honest decoding activation with the output $\perp$, matching the real decoder, which aborts on $\perp$. We call the *opening* the first corruption that exposes the decoder state: the corruption of $\mathcal{P}_{\mathsf{dcd}}$, or in the secret-key case of either party, since there $\mathsf{ek} = \mathsf{dk}$. In the public-key case, corrupting $\mathcal{P}_{\mathsf{ecd}}$ reveals only $\mathsf{ek} = \mathsf{pp}$ by erasure and is not an opening. A word is *close* if it is within distance δ of some issued codeword and *fresh* otherwise.

($\Rightarrow$) Suppose Π_{PRC} UC-realizes $\mathcal{F}_{\mathsf{PRC}}$ against adaptive corruptions with decoder-state exposure, and let $\mathcal{S}_{\mathsf{PRC}}$ be the corresponding ideal-process adversary. An adversary that never corrupts is in particular non-adaptive, so Theorems 1 and 2 give ideal security in the secret-key case and IND-CCA security in the public-key

[6] In the public-key case, the ideal-process adversary generates the key pair with $\mathtt{KGen}$ and decodes fresh words with the real decoder.

[7] We assume that dist is a metric under which two independent uniform words of $\Sigma^\star$ are within distance 2δ only with negligible probability, so that, as for honest codewords (Fig. 1), queries close to dummy codewords of different messages occur only with negligible probability.

case. We extract $\mathsf{NC\text{-}PRC} = (\mathtt{SimKGen}, \mathtt{SimCode}, \mathtt{OpenDec})$ by internally running the ideal execution of $\mathcal{F}_{\mathsf{PRC}}$ with $\mathcal{S}_{\mathsf{PRC}}$ in a fresh session, with both $\mathcal{P}_{\mathsf{ecd}}$ and $\mathcal{P}_{\mathsf{dcd}}$ initially honest.

$\mathsf{NC\text{-}PRC}.\mathtt{SimKGen}(1^\lambda)$ starts this internal ideal execution and emulates the $\langle \mathtt{KeyGen} : \cdot \rangle$ activation: in the public-key case $\mathcal{S}_{\mathsf{PRC}}$ returns a public encoding key ek and $\mathsf{NC\text{-}PRC}.\mathtt{SimKGen}(\cdot)$ sets $\mathsf{pp} := \mathsf{ek}$; in the secret-key case $\mathsf{pp} := \bot$. It outputs $(\mathsf{pp}, \mathsf{st}, \tau_k)$, where st records the internal state of the simulated ideal execution and τ_k is a key-opening handle identifying the simulated $\langle \mathtt{KeyGen} : \cdot \rangle$ activation; both are held for $\mathsf{NC\text{-}PRC}.\mathtt{OpenDec}(\cdot)$, which later resumes this execution. In the public-key case, $\mathcal{S}_{\mathsf{PRC}}$ generates $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathtt{KGen}(1^\lambda)$ by assumption, so st contains this dk.

$\mathsf{NC\text{-}PRC}.\mathtt{SimCode}(\mathsf{pp})$ emulates one honest encoding activation as seen by $\mathcal{S}_{\mathsf{PRC}}$: it samples $c \leftarrow\!\!\$\ \Sigma^\star$, as $\mathcal{F}_{\mathsf{PRC}}$ does, and returns (c, τ_c), where τ_c records the index of the simulated $\langle \mathtt{Encode} : \cdot \rangle$ activation that produced c. Hence $\mathsf{NC\text{-}PRC}$ is compatible with $\mathcal{F}_{\mathsf{PRC}}$. Since $\mathsf{NC\text{-}PRC}.\mathtt{SimCode}(\cdot)$ does not carry the execution state itself, the handles τ_c are what allow the transcript $\mathcal{T} = \{(m_i, c_i, \tau_{c_i})\}_i$ to be aligned with the internal execution held in st.

Given $\mathcal{T}$, the algorithm $\mathsf{NC\text{-}PRC}.\mathtt{OpenDec}(\mathsf{st}, \tau_k, \mathcal{T})$ resumes the internal execution identified by (st, τ_k), interprets the dummy codewords c_i as the issued pairs (m_i, c_i) via their handles τ_{c_i}, and corrupts $\mathcal{P}_{\mathsf{dcd}}$ in the public-key case and $\mathcal{P}_{\mathsf{ecd}}$ in the secret-key case. Upon corruption, $\mathcal{F}_{\mathsf{PRC}}$ discloses the issued table to $\mathcal{S}_{\mathsf{PRC}}$, and $\mathsf{NC\text{-}PRC}.\mathtt{OpenDec}(\cdot)$ outputs the state $\mathsf{dk}^\star$ contained in the fabricated corruption view, which is syntactically valid by the decoder-state exposure model.

Decoder-opening indistinguishability. An IND-OPEN adversary $\mathcal{A}_{ind}$ yields a UC environment $\mathcal{Z}$ that implements $\mathcal{O}_{\mathsf{Encode}}(m)$ by activating the honest encoder on $\langle \mathtt{Encode} : \mathsf{ecd}, \mathsf{sid}, m \rangle$, $\mathcal{O}_{\mathsf{Decode}}(x)$ in the public-key case by returning $\bot$ if x is close and otherwise activating the honest decoder on $\langle \mathtt{Decode} : \mathsf{dcd}, \mathsf{sid}, x \rangle$, and $\mathcal{O}_{\mathsf{CorruptDec}}$ by adaptively corrupting $\mathcal{P}_{\mathsf{dcd}}$ in the public-key case and $\mathcal{P}_{\mathsf{ecd}}$ in the secret-key case, handing the exposed state to $\mathcal{A}_{ind}$. After the corruption, $\mathcal{Z}$ answers $\mathcal{O}_{\mathsf{Encode}}$ and $\mathcal{O}_{\mathsf{Decode}}$ with $\bot$ itself, as the game does, and it outputs the bit of $\mathcal{A}_{ind}$. In the real execution, honest codewords are $\mathsf{PRC}.\mathtt{Encode}(\mathsf{ek}, m)$, the honest decoder answers $\mathsf{PRC}.\mathtt{Decode}(\mathsf{dk}, x)$, and the exposed state is dk, which is the real branch of IND-OPEN. In the ideal execution, honest codewords are uniform strings distributed as the outputs of $\mathsf{NC\text{-}PRC}.\mathtt{SimCode}(\cdot)$, a fresh word is delegated to $\mathcal{S}_{\mathsf{PRC}}$, which answers $\mathsf{PRC}.\mathtt{Decode}(\mathsf{dk}, x)$ for the key dk in st, and the exposed state is that of $\mathsf{NC\text{-}PRC}.\mathtt{OpenDec}(\cdot)$, which is the simulated branch. Hence any non-negligible advantage of $\mathcal{A}_{ind}$ contradicts the assumed realization.

δ-Opening robustness (secret-key case). Let $\mathcal{A}_{rob} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary against δ-opening robustness. The environment $\mathcal{Z}$ runs $\mathcal{A}_1$, answering its encoding queries through the honest encoder, then corrupts only $\mathcal{P}_{\mathsf{ecd}}$, obtains the state K, and runs $\mathcal{A}_2$ on K to obtain x. It then activates the still honest decoder on $\langle \mathtt{Decode} : \mathsf{dcd}, \mathsf{sid}, x \rangle$, receives y, and outputs 1 iff x is close to transcript codewords of a unique message and $y \neq \mathsf{PRC}.\mathtt{Decode}(K, x)$. In the real execution, $K = \mathsf{ek} = \mathsf{dk}$ and the honest decoder answers $y = \mathsf{PRC}.\mathtt{Decode}(\mathsf{dk}, x)$, so $\mathcal{Z}$ never outputs 1. In the ideal execution, $K = \mathsf{dk}^\star$ is the output of $\mathsf{NC\text{-}PRC}.\mathtt{OpenDec}(\cdot)$,

and since $\mathcal{P}_{\mathsf{dcd}}$ is honest, $\mathcal{F}_{\mathsf{PRC}}$ answers a query close to issued codewords of a unique message m with $y = m$ from the table; hence $\mathcal{Z}$ outputs 1 exactly when $\mathcal{A}_{rob}$ violates δ-opening robustness. By the realization, this happens with negligible probability.

($\Leftarrow$) Let NC-PRC be as in the statement. We define the ideal-process adversary $\mathcal{S}_{\mathsf{PRC}}$, which keeps a list $\mathcal{L}$ of handles.

- Upon $\langle \mathtt{KeyGen} : \mathsf{sid} \rangle$, it runs $(\mathsf{pp}, \mathsf{st}, \tau_k) \leftarrow \mathsf{NC\text{-}PRC}.\mathtt{SimKGen}(1^\lambda)$ and, in the public-key case, returns $\mathsf{ek} := \mathsf{pp}$.
- Upon an encoding notification while both parties are honest, it appends the index of the activation to $\mathcal{L}$.
- Upon a delegated fresh decoding query x before the opening, which occurs only in the public-key case, it returns $\mathsf{PRC}.\mathtt{Decode}(\mathsf{dk}, x)$ for the key dk in st.
- At the opening, it receives $\mathsf{Issued}[\mathsf{sid}]$, forms $\mathcal{T}$ by attaching to each issued pair its handle in $\mathcal{L}$, and reveals $K := \mathsf{dk}^\star \leftarrow \mathsf{NC\text{-}PRC}.\mathtt{OpenDec}(\mathsf{st}, \tau_k, \mathcal{T})$ as the state of the corrupted party, and as the state of the other party if it is corrupted later. Corrupting $\mathcal{P}_{\mathsf{ecd}}$ in the public-key case reveals only pp.
- Afterwards, it answers a delegated encoding of m with $\mathsf{PRC}.\mathtt{Encode}(K, m)$ in the secret-key case and with $\mathsf{PRC}.\mathtt{Encode}(\mathsf{pp}, m)$ in the public-key case, the latter also after a corruption of $\mathcal{P}_{\mathsf{ecd}}$ that is not an opening, and a delegated decoding of a word x for the honest decoder with $\mathsf{PRC}.\mathtt{Decode}(K, x)$. Activations of corrupted parties are handled as the adversary instructs, exactly as in the real execution.

Fix an environment $\mathcal{Z}$, and let $\mathbf{HYB}_0$ be the real execution $\mathbf{REAL}_{\Pi_{\mathsf{PRC}}, \mathcal{A}, \mathcal{Z}}$. In $\mathbf{HYB}_1$, before the opening, the honest decoder answers by the decoding rule of the ideal or random branch of Fig. 1 instead of running $\mathsf{PRC}.\mathtt{Decode}(\mathsf{dk}, \cdot)$: a close word with m if it is close to issued codewords of a unique message m and with $\perp$ if it is close to issued codewords of different messages, and a fresh word with $\perp$ in the secret-key case and with $\mathsf{PRC}.\mathtt{Decode}(\mathsf{dk}, \cdot)$ in the public-key case. The two hybrids are identical until the event Bad that some such answer differs from the real decoder. An adversary against ideal security or IND-CCA security runs $\mathcal{Z}$, answers honest encodings and honest decodings through its oracles, compares the answer to each decoding query with this rule applied to its own record of the transcript, and outputs 0 on the first mismatch and 1 at the opening or at the end. In the ideal or random branch, the decoding oracle answers by this very rule, so it never outputs 0; in the real branch, it outputs 0 exactly on Bad. Since the opening ends the simulation before any key is revealed, $\Pr[\mathsf{Bad}]$ is bounded by the advantage against ideal security or IND-CCA security.

We next build an IND-OPEN adversary $\mathcal{B}$ that runs $\mathcal{Z}$ and emulates $\mathbf{HYB}_1$. It hands pp to $\mathcal{Z}$ as the encoding key in the public-key case. Before the opening, it answers an encoding activation of the honest encoder while both parties are honest by $\mathcal{O}_{\mathsf{Encode}}$, an encoding requested after a corruption of $\mathcal{P}_{\mathsf{ecd}}$ in the public-key case by $\mathsf{PRC}.\mathtt{Encode}(\mathsf{pp}, \cdot)$, and a decoding activation of the honest decoder by the table rule on its record $\mathcal{T}$ of the codewords returned by $\mathcal{O}_{\mathsf{Encode}}$ if the word is close, and otherwise by $\mathcal{O}_{\mathsf{Decode}}$ in the public-key case and with $\perp$ in the

secret-key case. At the opening, it calls $\mathcal{O}_{\mathsf{CorruptDec}}$, obtains K, and reveals it. Afterwards, it answers encodings of the honest encoder with $\mathsf{PRC.Encode}(K,\cdot)$ in the secret-key case and $\mathsf{PRC.Encode}(\mathsf{pp},\cdot)$ in the public-key case, and a decoding query x of the honest decoder, which then exists only in the secret-key case with $\mathcal{P}_{\mathsf{ecd}}$ corrupted, by the table rule if x is close and with $\mathsf{PRC.Decode}(K,x)$ otherwise. It outputs the bit of $\mathcal{Z}$.

In the simulated branch, the dummy codewords are uniform by compatibility, close words are answered by the table rule of $\mathcal{F}_{\mathsf{PRC}}$, fresh words are rejected in the secret-key case and decoded under the key dk in st in the public-key case, the revealed state is that of $\mathsf{NC\text{-}PRC.OpenDec}(\cdot)$, and the answers after the opening are those of $\mathcal{F}_{\mathsf{PRC}}$ and $\mathcal{S}_{\mathsf{PRC}}$. Hence $\mathcal{B}$ emulates $\mathbf{IDEAL}_{\mathcal{F}_{\mathsf{PRC}},\mathcal{S}_{\mathsf{PRC}},\mathcal{Z}}$ exactly. In the real branch, the codewords, the decoding answers before the opening, the revealed state $K=\mathsf{dk}$, and the encodings and fresh decodings after the opening are those of $\mathbf{HYB}_1$. The emulation thus coincides with $\mathbf{HYB}_1$ unless the event Bad' occurs: after the opening, some decoding query x of the honest decoder is close and the table answer differs from $\mathsf{PRC.Decode}(K,x)$. Hence

$$\begin{aligned}&\left|\Pr[\mathbf{REAL}_{\Pi_{\mathsf{PRC}},\mathcal{A},\mathcal{Z}}=1]-\Pr[\mathbf{IDEAL}_{\mathcal{F}_{\mathsf{PRC}},\mathcal{S}_{\mathsf{PRC}},\mathcal{Z}}=1]\right|\\&\qquad\le \Pr[\mathsf{Bad}]+\mathbf{Adv}^{\text{IND-OPEN}}_{\mathsf{PRC},\mathsf{NC\text{-}PRC}}(\mathcal{B})+\Pr_{b=0}[\mathsf{Bad}'],\end{aligned}$$

where $\Pr_b$ is taken in the branch b of the game run by $\mathcal{B}$. In the public-key case, no honest decoder remains after the opening, so $\Pr_{b=0}[\mathsf{Bad}']=0$.

In the secret-key case, Bad' is efficiently checkable from K, $\mathcal{T}$ and the queries. Let $\mathcal{B}'$ run $\mathcal{B}$ and output 1 iff Bad' occurs, so that $\Pr_{b=0}[\mathsf{Bad}']\le\Pr_{b=1}[\mathsf{Bad}']+\mathbf{Adv}^{\text{IND-OPEN}}_{\mathsf{PRC},\mathsf{NC\text{-}PRC}}(\mathcal{B}')$. In the simulated branch, Bad' requires either a query x close to codewords of a unique message m with $\mathsf{PRC.Decode}(\mathsf{dk}^\star,x)\neq m$, or a query close to two dummy codewords of different messages. The first is a violation of δ-opening robustness by the adversary whose first stage runs $\mathcal{B}$ up to the opening, answering decoding queries by the table rule and with $\perp$, and whose second stage continues the emulation and outputs the first such query. The second requires two of the uniform dummy codewords to be within distance 2δ, by the triangle inequality, which happens only with negligible probability by the assumption in the statement of Theorem 3 and a union bound. Hence $\Pr_{b=1}[\mathsf{Bad}']$ is negligible, and so is the distinguishing advantage of $\mathcal{Z}$. □

C.4 Proof of Lemma 1

Lemma 1. *Let* $\mathsf{PRC}=(\mathtt{KGen},\mathtt{Encode},\mathtt{Decode})$ *be a* δ*-robust pseudorandom code. If* PRC *has* $(\mathbf{m},q)$*-targeted low opening capacity for some* $q\in\mathbb{N}$ *and* $\mathbf{m}\in\Sigma^q$*, then* PRC *does not admit a decoder-non-committing pseudorandom code compatible with* $\mathcal{F}_{\mathsf{PRC}}$ *that satisfies* δ*-opening robustness or* IND-OPEN*.*

Proof. We first rule out δ-opening robustness. Assume, toward contradiction, that PRC admits a compatible decoder-non-committing pseudorandom code $\mathsf{NC\text{-}PRC}=(\mathtt{SimKGen},\mathtt{SimCode},\mathtt{OpenDec})$ that satisfies δ-opening robustness.

Let $\mathbf{m}=(m_1,\dots,m_q)$ be the message sequence for which PRC has $(\mathbf{m},q)$-targeted low opening capacity. We run $(\mathsf{pp},\mathsf{st},\tau_k)\leftarrow\mathsf{NC\text{-}PRC.SimKGen}(1^\lambda)$. Then,

for every $i \in [q]$, obtain $(c_i, \tau_{c_i}) \leftarrow \mathsf{NC\text{-}PRC.SimCode}(\mathsf{pp})$ for message m_i, where, by compatibility, $\mathsf{NC\text{-}PRC.SimCode}(\cdot)$ simply samples the dummy codewords $c_1, \ldots, c_q$ as independent uniform elements from the codeword space. Then we form the opening transcript $\mathcal{T} := \{(m_i, c_i, \tau_{c_i})\}_{i \in [q]}$ and compute $\mathsf{dk}^\star \leftarrow \mathsf{NC\text{-}PRC.OpenDec}(\mathsf{st}, \tau_k, \mathcal{T})$ where $\mathsf{dk}^\star$ is syntactically valid by the definition of the opening algorithm.

We claim that, except with negligible probability, $\mathtt{Decode}(\mathsf{dk}^\star, c_i) = m_i$ for all $i \in [q]$. Indeed, suppose not. Then with non-negligible probability there exists some $i \in [q]$ such that $\mathtt{Decode}(\mathsf{dk}^\star, c_i) \neq m_i$. Consider the opening-robustness adversary that first queries the simulated encoder on the messages $m_1, \ldots, m_q$, receives $c_1, \ldots, c_q$, obtains the opened decoder state $\mathsf{dk}^\star$, and then outputs $x = c_i$ for such a failing index i.

For $x = c_i$, the transcript contains (m_i, c_i, τ_{c_i}), and hence x is within distance δ of a codeword prescribed to decode to m_i. Except for the negligible event that c_i lies in an ambiguous overlap with another transcript codeword carrying a different message, m_i is the unique message in $\mathsf{SimNear}^{\delta}_{\mathcal{T}}(c_i)$. Therefore the event $\mathtt{Decode}(\mathsf{dk}^\star, c_i) \neq m_i$ violates δ-opening robustness. Hence, except with negligible probability, the opened state satisfies $\mathtt{Decode}(\mathsf{dk}^\star, c_i) = m_i$ for every $i \in [q]$.

Thus, with non-negligible probability over independent uniform codewords $c_1, \ldots, c_q$, there exists a syntactically valid decoder state $\mathsf{dk}^\star$ such that $\forall i \in [q]$, $\mathtt{Decode}(\mathsf{dk}^\star, c_i) = m_i$. This is exactly the bad event ruled out by $(\mathbf{m}, q)$-targeted low opening capacity. We obtain a contradiction.

We next rule out IND-OPEN for any compatible NC-PRC. Let $\mathcal{A}$ query $\mathcal{O}_{\mathsf{Encode}}$ on $m_1, \ldots, m_q$, receive $c_1, \ldots, c_q$, call $\mathcal{O}_{\mathsf{CorruptDec}}$ to obtain a state K, and output 1 iff $\mathtt{Decode}(K, c_i) = m_i$ for all $i \in [q]$. In the real branch, $K = \mathsf{dk}$ and each c_i is an honest encoding of m_i; the adversary that outputs $x = c_i$ for a random $i \in [q]$ shows that the output is 1 except with probability at most q times the advantage against δ-robustness, since $\mathsf{dist}(c_i, c_i) = 0 < \delta$. In the simulated branch, $c_1, \ldots, c_q$ are independent uniform strings by compatibility, so by $(\mathbf{m}, q)$-targeted low opening capacity no syntactically valid K decodes all of them to $\mathbf{m}$ except with negligible probability. Hence $\mathcal{A}$ has non-negligible advantage. Therefore PRC does not admit a decoder-non-committing pseudorandom code compatible with $\mathcal{F}_{\mathsf{PRC}}$ that satisfies either property. □

C.5 Proof of Proposition 4

Proposition 4. *Let all obfuscated circuits in* $\mathsf{PRC}_0^{\mathsf{sk}}$ *and* $\mathsf{NC\text{-}PRC}_0^{\mathsf{sk}}$ *be padded to a common size* $s(\lambda, Q)$ *for an a-priori bound* Q *on openings. Suppose* pPRF *has perfect punctured correctness and pseudorandomness at punctured points,* iO *satisfies indistinguishability,* $|\mathcal{V}|$ *is superpolynomial, and the distance between* $\mathsf{ECC.Encode}(w) \oplus e$ *for* $w \leftarrow\!\!\$\ \Sigma$, $e \leftarrow\!\!\$\ \chi$ *and a uniform* $c \leftarrow\!\!\$\ \Sigma^\star$ *is within* $\epsilon(\lambda)$. *Then for every PPT adversary* $\mathcal{A}$ *making at most* $q \leq Q$ *encoding queries, all prior to its corruption query, there exist PPT adversaries* $\mathcal{B}_1, \mathcal{B}_2$ *such that*

$$\mathbf{Adv}^{\text{IND-OPEN}}_{\mathsf{PRC}_0^{\mathsf{sk}}, \mathsf{NC\text{-}PRC}_0^{\mathsf{sk}}}(\mathcal{A}) \;\leq\; \mathbf{Adv}^{\text{IND}}_{\mathsf{iO}}(\mathcal{B}_1) + \mathbf{Adv}^{\text{PRND}}_{\mathsf{pPRF}}(\mathcal{B}_2) + \frac{q^2}{2|\mathcal{V}|} + q \cdot \epsilon(\lambda).$$

Proof. Let $\mathbf{HYB}_0$ be the real branch ($b = 0$) of IND-OPEN, answering the i-th query by $c_i \leftarrow \mathsf{ECC.Encode}(v_i \| P(v_i)) \oplus e_i$ for $v_i \leftarrow\!\!\$\ \mathcal{V}$ and corruption by $\mathsf{dk} = P$; write $V \coloneqq \{v_i\}_{i\in[q]}$.

$\mathbf{HYB}_1 \approx \mathbf{HYB}_0$*:* Upon corruption, return $\mathsf{iO}(\widehat{P})$, where $\widehat{P}$ is the program that hardwires $\mathsf{pPRF.Eval}(\mathsf{msk}, v)$ on each $v \in V$ and runs $\mathsf{pPRF.PuncEval}(\mathsf{psk}, \cdot)$ elsewhere, for $\mathsf{psk} \leftarrow \mathsf{pPRF.Puncture}(\mathsf{msk}, V)$. By perfect punctured correctness, $\widehat{P}$ and $\mathsf{pPRF.Eval}(\mathsf{msk}, \cdot)$ compute the same function on all of $\mathcal{V}$, and both circuits are padded to size $s(\lambda, Q)$. The reduction $\mathcal{B}_1$ samples msk and all game randomness, submits the two circuits to its iO challenger, embeds the returned obfuscation as the corruption response, and outputs $\mathcal{A}$'s bit, giving

$$|\Pr[\mathbf{HYB}_0 \Rightarrow 1] - \Pr[\mathbf{HYB}_1 \Rightarrow 1]| \le \mathbf{Adv}^{\mathrm{IND}}_{\mathsf{iO}}(\mathcal{B}_1).$$

$\mathbf{HYB}_2 \approx \mathbf{HYB}_1$*:* Replace each value $\mathsf{pPRF.Eval}(\mathsf{msk}, v)$ for $v \in V$ by a uniform $t_v \leftarrow\!\!\$\ \mathcal{T}$, both inside the codewords $c_i \leftarrow \mathsf{ECC.Encode}(v_i \| t_{v_i}) \oplus e_i$ and hardwired into the opened program. The reduction $\mathcal{B}_2 = (\mathcal{B}_{2,1}, \mathcal{B}_{2,2})$ lets $\mathcal{B}_{2,1}$ presample $v_1, \ldots, v_q \leftarrow\!\!\$\ \mathcal{V}$ and output the deduplicated set $S \coloneqq V$; then $\mathcal{B}_{2,2}$, on input $(\mathsf{psk}, \{y_v\}_{v\in V})$, answers the i-th query with $\mathsf{ECC.Encode}(v_i \| y_{v_i}) \oplus e_i$, responds to corruption with iO of the program hardwiring $\{(v, y_v)\}_{v\in V}$ over $\mathsf{pPRF.PuncEval}(\mathsf{psk}, \cdot)$, and outputs $\mathcal{A}$'s bit hence

$$|\Pr[\mathbf{HYB}_1 \Rightarrow 1] - \Pr[\mathbf{HYB}_2 \Rightarrow 1]| \le \mathbf{Adv}^{\mathrm{PRND}}_{\mathsf{pPRF}}(\mathcal{B}_2).$$

$\mathbf{HYB}_3 \approx \mathbf{HYB}_2$*:* The hybrid $\mathbf{HYB}_3$ is the simulated branch ($b = 1$): each $c_i \leftarrow\!\!\$\ \Sigma^\star$ is sampled by $\mathsf{NC\text{-}PRC}_0^{\mathsf{sk}}.\mathsf{SimCode}(\cdot)$, and corruption is answered by $\mathsf{dk}^\star \leftarrow \mathsf{NC\text{-}PRC}_0^{\mathsf{sk}}.\mathsf{OpenDec}(\mathsf{st}, \mathcal{T})$.

We first argue that $\mathsf{NC\text{-}PRC}_0^{\mathsf{sk}}.\mathsf{OpenDec}(\cdot)$, applied to the codewords of $\mathbf{HYB}_2$, yields the view of $\mathcal{A}$ in $\mathbf{HYB}_2$. Since each noise term e_i lies within the decoding radius δ, $\mathsf{NC\text{-}PRC}_0^{\mathsf{sk}}.\mathsf{OpenDec}(\cdot)$ recovers exactly the encoded pair $\mathsf{ECC.Decode}(c_i) = v_i \| t_{v_i}$. So conditioned on $v_1, \ldots, v_q$ being pairwise distinct, which fails with probability at most $q^2/2|\mathcal{V}|$, it punctures at the same points and hardwires the same pairs as $\mathbf{HYB}_2$. The only discrepancy is that $\mathsf{NC\text{-}PRC}_0^{\mathsf{sk}}.\mathsf{OpenDec}(\cdot)$ uses the master key stored in st, freshly sampled by $\mathsf{NC\text{-}PRC}_0^{\mathsf{sk}}.\mathsf{SimKGen}(\cdot)$, whereas $\mathbf{HYB}_2$ uses the game's msk; this is immaterial, since the hardwired values t_{v_i} are uniform and independent of the master key, and the corruption response depends on it only through the punctured key and these hardwired values, so both induce the same distribution on the view.

The two hybrids thus differ only in the codeword distributions fed to the same post-processing: in $\mathbf{HYB}_2$ each $c_i = \mathsf{ECC.Encode}(w_i) \oplus e_i$ has an independent uniform input $w_i = v_i \| t_{v_i} \leftarrow\!\!\$\ \Sigma$, while in $\mathbf{HYB}_3$ each c_i is uniform, so replacing them one sample at a time costs $q \cdot \epsilon(\lambda)$ by hypothesis. Hence

$$|\Pr[\mathbf{HYB}_2 \Rightarrow 1] - \Pr[\mathbf{HYB}_3 \Rightarrow 1]| \ \le\ \frac{q^2}{2|\mathcal{V}|} + q \cdot \epsilon(\lambda).$$

Therefore, by combining the hybrids above, we have that

$$\mathbf{Adv}^{\text{IND-OPEN}}_{\mathsf{PRC}_0^{\mathsf{sk}},\mathsf{NC\text{-}PRC}_0^{\mathsf{sk}}}(\mathcal{A}) \;\le\; \mathbf{Adv}^{\text{IND}}_{\mathsf{iO}}(\mathcal{B}_1) + \mathbf{Adv}^{\text{PRND}}_{\mathsf{pPRF}}(\mathcal{B}_2) + \frac{q^2}{2|\mathcal{V}|} + q \cdot \epsilon(\lambda).$$

□

C.6 Proof of Proposition 6

Proposition 6. *Assume that* SPHF *satisfies projectivity, sparsity with bound* $\epsilon_{\text{sp}}(\lambda)$*, subset membership indistinguishability, and* $(q+1)$*-smoothness with distance* $\epsilon_{\text{sm}}(\lambda)$*; that* $|U|$ *is superpolynomial; and that the statistical distance between* $\mathsf{ECC.Encode}(w) \oplus e$*, for independent* $w \leftarrow\!\$\ \Sigma$ *and* $e \leftarrow\!\$\ \chi$*, and a uniform* $c \leftarrow\!\$\ \Sigma^\star$ *is at most* $\epsilon(\lambda)$*. Assume moreover* δ*-tightness: for* $c \leftarrow\!\$\ \Sigma^\star$*, every PPT algorithm given* c *outputs some* x *with* $\mathsf{dist}(x,c) \ge \delta$ *and* $\mathsf{ECC.Decode}(x) = \mathsf{ECC.Decode}(c) \neq \perp$ *with probability at most* $\epsilon_{\text{tg}}(\lambda)$*. Then for every PPT adversary* $\mathcal{A}$ *making at most* q *encoding queries and at most* Q_d *decoding queries, all prior to its corruption query, there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\text{IND-OPEN}}_{\mathsf{PRC}_0^{\mathsf{pk}},\mathsf{NC\text{-}PRC}_0^{\mathsf{pk}}}(\mathcal{A}) \;\le\; q \cdot \mathbf{Adv}^{\text{IND-SM}}_{\mathsf{SPHF}}(\mathcal{B}) + \epsilon_{\text{sm}} + q \cdot \epsilon_{\text{sp}} + \frac{q^2}{2|U|} + 2q \cdot \epsilon(\lambda) + q \cdot \epsilon_{\text{tg}} + Q_d \cdot \Big(3\epsilon_{\text{sm}} + \frac{2}{|\mathcal{H}|}\Big).$$

Proof. Since the codewords do not depend on the messages of a zero-bit code, we presample all values used by the at most q encoding queries. For a word x, write $u_x \| h_x \leftarrow \mathsf{ECC.Decode}(x)$, and call x *close* if $\mathsf{dist}(x, c_i) < \delta$ for some codeword c_i returned so far and *fresh* otherwise. A key K *accepts* x if $\mathtt{Decode}(K, x) = 1$, i.e., $u_x \| h_x \neq \perp$ and $\mathsf{SPHF.Hash}(K, u_x) = h_x$. The oracle $\mathcal{O}_{\mathsf{Decode}}$ rejects close words in both branches and answers a fresh word with $\mathtt{Decode}(\mathsf{hk}, x)$, for the real key hk in the real branch and for the key hk in st in the simulated branch.

Let $\mathbf{HYB}_0$ be the real branch ($b = 0$) of IND-OPEN, answering the i-th encoding query by $c_i \leftarrow \mathsf{ECC.Encode}\big(u_i \| \mathsf{SPHF.Proj}(\mathsf{hp}, u_i, w_i)\big) \oplus e_i$ for $(u_i, w_i) \leftarrow\!\$\ \mathcal{R}$, decoding queries by $\mathtt{Decode}(\mathsf{hk}, \cdot)$, and corruption by $\mathsf{dk} = \mathsf{hk}$.

$\mathbf{HYB}_1 \approx \mathbf{HYB}_0$*:* Compute each hash as $h_i \leftarrow \mathsf{SPHF.Hash}(\mathsf{hk}, u_i)$ instead of $\mathsf{SPHF.Proj}(\mathsf{hp}, u_i, w_i)$. By projectivity the two values coincide on $\mathcal{R}$, so the hybrids are identical (up to the negligible projectivity failure, which we absorb).

$\mathbf{HYB}_2 \approx \mathbf{HYB}_1$*:* Sample each statement uniformly, $u_i \leftarrow\!\$\ U$, discarding the witness, and keep $h_i \leftarrow \mathsf{SPHF.Hash}(\mathsf{hk}, u_i)$. The reduction $\mathcal{B}$ samples $\mathsf{hk} \leftarrow \mathtt{HashKG}(1^\lambda)$ itself, embeds its challenge element as $u_{i^\star}$ for a hybrid index $i^\star \leftarrow\!\$\ [q]$, computes all hashes, answers decoding queries and corruption with hk locally, and outputs $\mathcal{A}$'s bit; a standard hybrid over the q queries gives

$$|\Pr[\mathbf{HYB}_1 \Rightarrow 1] - \Pr[\mathbf{HYB}_2 \Rightarrow 1]| \le q \cdot \mathbf{Adv}^{\text{IND-SM}}_{\mathsf{SPHF}}(\mathcal{B}).$$

From now on, we condition on the statements $u_1, \ldots, u_q$ being pairwise distinct and outside L, which fails with probability at most $q^2/2|U| + q \cdot \epsilon_{\text{sp}}$.

$\mathbf{HYB}_3 \approx \mathbf{HYB}_2$*:* Sample uniform $h_1, \ldots, h_q \leftarrow\!\$\ \mathcal{H}$ and a key K' uniformly from the fiber $\mathcal{K}^\star$ of $(\mathsf{hp}, \{(u_i, h_i)\}_i)$, and use K' in place of hk both for fresh decoding queries and for corruption. In $\mathbf{HYB}_2$, the game is a function of $(\mathsf{hp}, \{(u_i, \mathsf{SPHF.Hash}(\mathsf{hk}, u_i))\}_i, \mathsf{hk})$, and in $\mathbf{HYB}_3$ it is the same function of $(\mathsf{hp}, \{(u_i, h_i)\}_i, K')$. By q-smoothness, $(\mathsf{hp}, \{\mathsf{SPHF.Hash}(\mathsf{hk}, u_i)\}_i)$ is within distance ϵ_{sm} of $(\mathsf{hp}, \{h_i\}_i)$, and since hk is uniform over $\mathcal{K}_{\mathsf{hk}}$, its conditional distribution given this tuple is uniform over the fiber, as is that of K'. Hence the two hybrids differ by at most ϵ_{sm}.

$\mathbf{HYB}_4 \approx \mathbf{HYB}_3$*:* Answer fresh decoding queries with the key hk sampled at the start, which in $\mathbf{HYB}_3$ is used only through hp, while corruption still reveals K'. The two hybrids are identical until the event Bad that some fresh query x with $u_x \notin L$ is accepted by hk or by K': for $u_x \in L$, both keys have projection key hp, so they agree by projectivity. Until Bad, every fresh query with $u_x \notin L$ is rejected in both hybrids, so $\mathcal{A}$'s view depends on hk only through hp and on K' only through $(\mathsf{hp}, \{(u_i, h_i)\}_i)$. Consider a fresh query x with $u_x \notin L$.

- If $u_x \| h_x = u_i \| h_i$ for some i, then $\mathsf{dist}(x, c_i) \geq \delta$ and $\mathsf{ECC.Decode}(x) = \mathsf{ECC.Decode}(c_i)$. Embedding the challenge of δ-tightness as c_i for a random i, at a cost of ϵ for replacing the honest codeword by a uniform one, this happens for some query with probability at most $q \cdot (\epsilon_{\mathrm{tg}} + \epsilon)$.
- Otherwise, $\mathsf{SPHF.Hash}(\mathsf{hk}, u_x)$ is within ϵ_{sm} of uniform given hp, and $\mathsf{SPHF.Hash}(K', u_x)$ is within $2\epsilon_{\mathrm{sm}}$ of uniform given $(\mathsf{hp}, \{h_i\}_i)$ by q- and $(q+1)$-smoothness, since $u_x \neq u_i$ or $h_x \neq h_i = \mathsf{SPHF.Hash}(K', u_i)$. Hence x is accepted by either key with probability at most $3\epsilon_{\mathrm{sm}} + 2/|\mathcal{H}|$.

By a union bound over the at most Q_d decoding queries,

$$|\Pr[\mathbf{HYB}_3 \Rightarrow 1] - \Pr[\mathbf{HYB}_4 \Rightarrow 1]| \leq q \cdot (\epsilon_{\mathrm{tg}} + \epsilon) + Q_d \cdot \Big(3\epsilon_{\mathrm{sm}} + \frac{2}{|\mathcal{H}|}\Big).$$

$\mathbf{HYB}_5 \approx \mathbf{HYB}_4$*:* The simulated branch ($b = 1$): each $c_i \leftarrow\!\$\ \Sigma^\star$ is sampled by $\mathsf{NC\text{-}PRC}_0^{\mathsf{pk}}.\mathsf{SimCode}(\cdot)$, fresh decoding queries are answered with $\mathsf{Decode}(\mathsf{hk}, \cdot)$ for the key hk in st, and corruption is answered by $\mathsf{NC\text{-}PRC}_0^{\mathsf{pk}}.\mathsf{OpenDec}(\mathsf{st}, \mathcal{T})$. Since each noise term e_i lies within the decoding radius δ, in $\mathbf{HYB}_4$ the algorithm $\mathsf{NC\text{-}PRC}_0^{\mathsf{pk}}.\mathsf{OpenDec}(\cdot)$ recovers exactly the encoded pairs (u_i, h_i) and samples $\mathsf{hk}^\star$ from the same fiber as K'. Hence both hybrids apply the same post-processing, which uses $(\mathsf{hk}, \mathsf{hp})$ and the codewords, to codewords that are $c_i = \mathsf{ECC.Encode}(w_i) \oplus e_i$ for independent uniform $w_i = u_i \| h_i \leftarrow\!\$\ \Sigma$ in $\mathbf{HYB}_4$ and uniform in $\mathbf{HYB}_5$. Replacing them one sample at a time costs $q \cdot \epsilon(\lambda)$, whence

$$|\Pr[\mathbf{HYB}_4 \Rightarrow 1] - \Pr[\mathbf{HYB}_5 \Rightarrow 1]| \leq q \cdot \epsilon(\lambda).$$

Combining the hybrids above gives the claimed bound. □