Ideal Pseudorandom Codes

Omar Alrabiah[*] Prabhanjan Ananth[†] Miranda Christ[‡] Yevgeniy Dodis[§]
Sam Gunn[¶]

November 8, 2024

Abstract

Pseudorandom codes are error-correcting codes with the property that no efficient adversary can distinguish encodings from uniformly random strings. They were recently introduced by Christ and Gunn [CRYPTO 2024] for the purpose of watermarking the outputs of randomized algorithms, such as generative AI models. Several constructions of pseudorandom codes have since been proposed, but none of them are robust to error channels that depend on previously seen codewords. This stronger kind of robustness is referred to as *adaptive robustness*, and it is important for meaningful applications to watermarking.

In this work, we show the following.

- **Adaptive robustness**: We show that the pseudorandom codes of Christ and Gunn are adaptively robust, resolving a conjecture posed by Cohen, Hoover, and Schoenbach [S&P 2025]. Our proof involves several new ingredients, combining ideas from both cryptography and coding theory and taking hints from the analysis of Boolean functions.
- **Ideal security**: We define an *ideal pseudorandom code* as one which is indistinguishable from the ideal functionality, capturing both the pseudorandomness and robustness properties in one simple definition. We show that any adaptively robust pseudorandom code for single-bit messages can be bootstrapped to build an ideal pseudorandom code with linear information rate, under no additional assumptions.
- **CCA security**: In the setting where the encoding key is made public, we define a CCA-secure pseudorandom code in analogy with CCA-secure encryption. We show that any adaptively robust public-key pseudorandom code for single-bit messages can be used to build a CCA-secure pseudorandom code with linear information rate, in the random oracle model.

Together with the result of Christ and Gunn, it follows that there exist ideal pseudorandom codes assuming the $2^{O(\sqrt{n})}$-hardness of LPN. This extends to CCA security in the random oracle model. These results immediately imply stronger robustness guarantees for generative AI watermarking schemes, such as the practical quality-preserving image watermarks of Gunn, Zhao, and Song (2024).

[*] UC Berkeley. Email: oalrabiah@berkeley.edu
[†] UCSB. Email: prabhanjan@cs.ucsb.edu
[‡] Columbia University. Email: mchrist@cs.columbia.edu
[§] NYU. Email: dodis@cs.nyu.edu
[¶] UC Berkeley. Email: gunn@berkeley.edu

Contents

1 Introduction

Pseudorandom codes. Suppose that Alice wishes to send a message to Bob over some channel. The fields of cryptography and coding theory each address a different concern that Alice might have about the channel:

- if the channel is untrusted in the sense that there may be an eavesdropper, then we use encryption;
- if the channel is unreliable in the sense that information may be corrupted in transit, then we use error-correcting codes.

If the channel is both untrusted and unreliable, there is typically no issue with combining these techniques. That is, if Alice wishes only to hide the *content* of her message, then she can simply error correct an encryption of her message. However, this is insufficient if Alice *doesn't want the channel to even know that communication has occurred.*[1]

Concretely, imagine that Alice wishes for it to appear as if her transmissions are uniformly random. What Alice needs is then *robust, pseudorandom encryption* — transmissions should appear random, and decryption should function even if they are corrupted. Such an object is called a *pseudorandom code* (PRC).

More formally, a (secret-key) PRC is a keyed error correction scheme consisting of algorithms for encoding and decoding. To an adversary without knowledge of the key, an encoding oracle is computationally indistinguishable from an oracle that outputs a freshly random string on each query. With the secret key, one can decode codewords even after they are subjected to an error channel.[2]

To build a PRC, the basic results of cryptography and coding theory fall short. The central difficulty is that Alice's transmission must, on the one hand, appear *completely structure-less* to the channel, and on the other, appear *highly redundant* to Bob so that he can reliably recover the message even if the transmission is corrupted. Therefore pseudorandom codes force us to push the boundaries of both cryptography and coding theory, a tension that is reflected in the fact that all known constructions of PRCs have quasipolynomial-time distinguishing attacks [CG24, GM24, GG24].

Watermarking. PRCs are not only natural cryptographic and coding-theoretic objects, but also powerful tools for constructing watermarks for generative AI. See Section 1.2 for an explanation of how this works; for now let us just say that at a high level, this is because one can typically view generative AI models as approximate reparameterizations of "content" (e.g. images or text) into a space where the content has a natural distribution (e.g. normal or uniform). With this observation, [CG24, GM24] use PRCs to construct watermarks for language models that are *undetectable* in that the watermarked model is computationally indistinguishable from the original model (thereby ensuring that the watermark does not degrade generation quality), and *robust* in that the watermark tolerates a constant rate of errors. Prior text watermarks were either only undetectable but not robust [CGZ24], or robust but significantly altered the model's output distribution [ZALW24, KTHL24]. Similarly, the only known quality-preserving and robust watermarks for image generation models uses a PRC. This approach was demonstrated in practice by [GZS24].

These schemes essentially embed a PRC codeword in the space where content has a natural distribution; pseudorandomness of the PRC implies undetectability of the watermark, and robustness of the PRC implies robustness of the watermark. As we explain in Section 1.2, PRCs are in fact *equivalent* to undetectable and robust watermarks for language models.

[1] This kind of covert communication is the classic problem of *steganography*, but classic stateless steganographic techniques fail on unreliable channels.

[2] We are interested in error channels that introduce a *constant* rate of errors, and reserve the term "pseudorandom code" for schemes that handle such channels. This is the more practically relevant as well as theoretically interesting setting; it is easy to handle any sub-constant error rate using just one-way functions [GG24].

Oblivious vs adaptive robustness. While existing PRCs (and their corresponding watermarks) tolerate a high rate of errors, these errors must be made *obliviously*, i.e., by a memoryless channel. This falls short of the worst-case error model commonly considered in coding theory. Furthermore, errors introduced by realistic adversaries are non-oblivious: A realistic adversary can see multiple watermarked responses, and sometimes even query a detection oracle. It turns out that PRCs (and watermarks) that are highly robust to oblivious attacks may be easily removable by such adversaries, as demonstrated by an example in [CHS24].[3] In fact, this is not just a theoretical possibility. Several practical attacks leverage access to encoding and decoding oracles to remove or forge watermarks [JSV24, PHZS24].

Therefore it is of both practical and theoretical importance to have PRCs that are robust to non-oblivious adversaries. We say that a PRC is *adaptively robust* if it can handle adversaries who are given an encoding oracle (or key). We introduce *ideal security* and *CCA security* to handle cases where the adversary additionally queries a detection oracle. Adaptive robustness was previously studied in [CHS24], but they showed only that the watermark of [CGZ24] was adaptively robust to a very low (specifically, inverse security parameter) rate of errors. [CHS24] conjectured but did not prove that the PRC of [CG24] is adaptively robust.

This work. We prove that a slight modification of the PRC of [CG24] is adaptively robust to a *constant rate of substitutions*, resolving the conjecture of [CHS24]. That is, for any $\delta < 1/4$ (or $\delta < 1/2$ if we wish only to detect but not decode messages), we show that no adversary can produce an error of relative weight at most δ that causes the decoder to fail — even if the adversary is provided with the encoding key. We call this property *adaptive δ-robustness.* Our proof involves an interesting combination of techniques from cryptography and coding theory.

We then ask whether it is possible to achieve robustness to an adversary with access to both an encoding oracle *and* a decoding oracle. Rather than contribute to a growing list of separately defined properties, we take a principled approach and consider the *ideal functionality* of a PRC. This functionality dictates how the code should behave to any adversary, regardless of the adversary's goal (e.g., distinguishing codewords from random, or mauling codewords). We say that a PRC satisfies *ideal security* (or is an *ideal PRC*) if the real PRC encoding and decoding oracles are indistinguishable from this ideal functionality. This definition combines soundness, pseudorandomness, and adaptive robustness into one simple definition.

We show a generic and simple transformation from *any* adaptively robust PRC capable of encoding a single-bit message to an ideal PRC with linear information rate. Our transformation requires only a pseudorandom function, the existence of which is implied by that of a PRC. This transformation, applied to the adaptively robust PRCs discussed above, yields a construction of an ideal PRC.

Finally, we turn to the public-key setting, where both pseudorandomness and robustness are defined with respect to adversaries that have access not only to an encoding *oracle*, but an encoding *key*. While ideal security is the strongest possible definition in the secret-key setting, there is no obvious "ideal" security notion in the public-key setting. Instead, we define a strengthening of adaptive robustness that we call *CCA security* in analogy with CCA secure encryption. This definition generalizes standard CCA security for pseudorandom encryption. In the random oracle model, we show a generic transformation that builds a CCA-secure PRC from any adaptively robust public-key PRC.

We summarize our four definitions of PRC robustness in Table 1.

[3] Let $\mathcal{W}$ be a watermarked model. Consider another watermarked model $\mathcal{W}'$ that, given a prompt π, outputs $\mathcal{W}(\pi)$ if π is unwatermarked and outputs an *unwatermarked response* if π is watermarked. Then $\mathcal{W}'$ has the same oblivious robustness as $\mathcal{W}$, because an oblivious adversary cannot find a watermarked π; however, it is not at all robust to an adversary that makes just two queries to $\mathcal{W}'$: The adversary can simply obtain a watermarked response x from $\mathcal{W}'$, then query $\mathcal{W}'$ on x to obtain an unwatermarked response. The adversary doesn't even need to make any edits!

	No decoder access	**Decoder access**
Secret-key	Secret-key adaptive robustness	Ideal security
Public-key	Public-key adaptive robustness	CCA security

Table 1: The four definitions of robustness considered in this work. "Secret-key" means the adversary is given oracle access to the encoder and "public-key" means the adversary is given the encoding key. "No decoder access" and "decoder access" distinguish whether the adversary is additionally given oracle access to the decoder. We satisfy the "no decoder access" column in Theorems 1 and 2, ideal security in Theorem 3, and CCA security in Theorem 4.

1.1 Results

In this subsection we highlight our main results, which can be summarized follows: (a) We prove unconditionally that certain PRCs based on LDPC codes are (maximally) adaptively robust in both the secret- and public-key settings; (b) In the secret-key setting, we present a generic transformation converting any adaptively robust PRC into an ideal PRC; (c) In the public-key setting, we present a generic transformation converting any adaptively robust PRC into a CCA-secure PRC in the random oracle model.

In presenting our results, we use the term "δ-robust" to indicate that the attacker can adaptively corrupt up to a δ fraction of the codeword. By "zero-bit" PRC we mean a PRC that is only capable of encrypting a fixed message; by "single-bit" we mean that the PRC can encrypt 0 or 1.

Our first two results show essentially optimal robustness of the zero-bit LDPC-based PRC from [CG24], and our single-bit version of it. Any zero-bit PRC is at most $1/2$-adaptively robust, since such errors can completely randomize the codeword. Any single-bit PRC is at most $1/4$-adaptively robust, since an adversary can always construct a string that is $1/4$-close to two codewords encoding different messages (see the remark at the end of Section 2.3).

Theorem 1 (Adaptively robust zero-bit PRC)**.** *For any $\varepsilon > 0$, the public-key zero-bit pseudorandom code from [CG24] is adaptively $(1/2 - \varepsilon)$-robust for appropriate choice of parameters.*

Theorem 2 (Adaptively robust single-bit PRC)**.** *For any $\varepsilon > 0$, our public-key single-bit pseudorandom code is adaptively $(1/4 - \varepsilon)$-robust for appropriate choice of parameters.*

Our secret-key transformation is lightweight in that it makes no additional assumptions beyond the security of the underlying PRC,[4] and it preserves the quantitative level of robustness.

Theorem 3 (Ideal PRC)**.** *Suppose that there exists a secret-key single-bit pseudorandom code that is adaptively δ-robust. Then there exists a δ-robust ideal pseudorandom code with linear information rate.*

The result is somewhat worse for the public-key case, incurring a constant-factor loss in robustness and requiring the random oracle model.

Theorem 4 (CCA-secure PRC)**.** *Suppose that there exists a public-key single-bit pseudorandom code that is adaptively δ-robust. Then in the random oracle model, there exists an $\Omega(\delta)$-robust CCA-secure pseudorandom code with linear information rate.*

By applying Theorems 3 and 4 to the PRCs from [CG24], which are pseudorandom assuming the certain-subexponential hardness of LPN and adaptively δ-robust for $\delta < 1/4$ by Theorem 2, we have the following corollary.

Corollary 5. *Assuming the $2^{O(\sqrt{\lambda})}$-hardness of learning parity with noise (LPN), there exist separate linear-rate PRCs satisfying*

- *δ-robust ideal security for any $\delta < 1/4$, and*

[4] It makes use of a pseudorandom function, whose existence is implied by that of a PRC.

- *δ-robust CCA security for some $\delta = \Omega(1)$ in the random oracle model.*

While the above robustness is optimal for the ideal PRC result, we do not achieve the optimal robustness for the CCA-secure PRC result.

1.2 Relationship to watermarking

Christ and Gunn [CG24] presented a general template for using pseudorandom codes to watermark sufficiently high-entropy outputs of randomized algorithms. We recall that template here for reference.

Consider a randomized algorithm that, given an input x and a random seed r, generates an output y. Suppose that there exists a *randomness recovery* algorithm that, given y, outputs an approximation of the random seed r used to generate it. Randomness recovery algorithms indeed exist for generative models used in practice, and their approximation accuracy increases with the amount of entropy in the output distribution. The basic observation is that replacing r with an output of $\mathsf{PRC.Encode}(1)$ immediately yields a watermark. Pseudorandomness ensures that this replacement does not perceptibly alter the quality of the model. Error correction enables detection, with the content being considered watermarked if $\mathsf{PRC.Decode}(\mathsf{RandomnessRecovery}(y)) = 1$. This watermarking approach works for *any* randomized algorithm with a randomness recovery algorithm whose approximation error is within the PRC's tolerance.

Remark. *It is not a coincidence that randomness recovery algorithms exist in almost every generative AI framework. In some cases, randomness recovery can be viewed as an essential part of training. It is also useful to have randomness recovery in many independent generative AI applications, because it can allow for content manipulation in a more natural "latent space" where features correspond to higher-order features of the content.*

It is known that this framework yields an undetectable and robust watermark for large language models [CG24, GM24]. This framework has also been demonstrated experimentally by [GZS24], which builds a robust undetectable watermarking scheme for generative image models (in particular, latent diffusion models) using essentially the same PRC as we consider here.

Not only are pseudorandom codes useful for constructing robust undetectable watermarks, but they are also necessary in a formal sense. Consider a model that can be prompted to output a uniformly random binary string (e.g., one can ask a language model to output a random sequence of "apple" and "orange"). Suppose we can undetectably watermark this model, with robustness to a given error channel. This implies that on each query, the model outputs a fresh pseudorandom string such that the detector outputs True even if this string is subjected to this error channel. Soundness (i.e., the low false positive rate) of the watermark ensures that unrelated strings decode to False. Thus, the model with this fixed prompt is *exactly* an encoder for a zero-bit PRC, where the PRC decoder is the watermark detector.

PRC robustness and watermark robustness. Typical randomness recovery algorithms for generative AI are crucially imperfect, even on unedited generations. If one wishes to have a *robust* watermark, then randomness recovery will be still more inaccurate. Therefore, it is essential for applications to watermarking that we use a PRC with strong robustness. The watermark will be detectable as long as the composition of an adversary's modifications to the output, and any intrinsic error from randomness recovery, falls within the class of error channels tolerated by the PRC.

For example, a PRC with robustness to a constant rate of errors chosen by an adversary *that observes only the given output* translates to a watermark with robustness to an adversary making a single query to the generation algorithm. In practice, however, an adversary attempting to remove the watermark can make arbitrarily many generation queries, translating to a PRC error channel that observes many codewords. Therefore we need our PRC to be adaptively robust if we wish to handle such watermark removal attacks.

Similarly, a watermark that is robust to an adversary with access to a watermark detection oracle necessitates a PRC that is robust to an adversary with a decoding oracle. This robustness is satisfied by the ideal PRCs that we define here. Similarly, an adversary given a watermark detection oracle *and the watermark generation key* is handled by our CCA-secure PRCs.

On robustness to substitutions. In this work, we only consider PRCs with robustness to substitutions. This is sufficient for some watermarking applications, like those where the PRC is embedded in a *semantic* representation of the content, and general edits in the content itself translate to substitutions in the semantic space. This is the case, for instance, in the image watermark of [GZS24] because the models they consider map between images and a *latent space* consisting of fixed-length vectors. They embed a PRC codeword in this latent representation of the generated image, and their decoder maps the given image back to its latent representation before decoding the PRC. Since these latent vectors are of fixed size, modifications to the image translate to changes to components, but not insertions or deletions.

For other applications — especially watermarking language models — it can be useful to have a PRC with robustness to more general forms of edits. Random deletions were considered in [CG24], and oblivious deletions and insertions were considered in [GM24] (over a polynomial-sized alphabet). However, we note that both of these methods work *by reduction to the binary substitution channel*. Therefore improved results about substitution channels may translate to improved results for editing channels, although we do not investigate this here.

On the alphabet size. One can think of the watermark generation process as sampling each "component" of the output y to be correlated with the corresponding bit of the random seed r. The amount of entropy in a given component y_i determines how much signal from the corresponding seed bit r_i it contains. In order to plant one symbol of a PRC codeword in each component with significant signal, we require $\Omega(|\text{PRC alphabet}|)$ entropy per component.

For language models, these "components" might be the tokens of the response. Typical language models have on the order of one bit of entropy per word, necessitating a PRC with a constant-sized (ideally, binary) alphabet.[5] Other watermarking applications will also suffer from the use of a larger alphabet. Therefore, in this work we only consider *binary-alphabet* PRCs.

Related work. We have referred to most of the relevant existing works, particularly those about pseudorandom codes, throughout this introduction. For a more detailed discussion on related work on watermarking, see Appendix A.

2 Technical overview

2.1 Organization

Adaptive robustness, ideal security, and CCA security define robustness for a PRC under various adversarial models, as shown in Table 1. See Figure 1 for a visual representation of the structure of the paper.

We briefly recall the definition of a pseudorandom code in Section 2.2 (and Section 3.1).

In Section 2.3 (and Section 4), we show that a particular PRC construction based on LDPC codes is adaptively robust in the public-key setting. This immediately implies that the same construction is adaptively robust in the secret-key setting. This is the only section that is particular to any specific PRC construction; all of our results in later sections are are generic in the underlying PRC.

In Section 2.4 (and Section 5), we show that in the secret-key setting, any single-bit adaptively δ-robust PRC can be converted to a δ-robust ideal PRC with a linear information rate.

[5] While it is possible to increase the entropy per component by taking each component to be a sequence of k words, this harms robustness as changing one in every k words now changes every symbol of the underlying codeword.

Section 2.5 (and Section 6) mirrors the preceding section, but in the public-key setting. As is the case with standard encryption, in the public-key setting there is no longer a clear notion of "ideal security," but we nonetheless present a definition we call "CCA security" in analogy with CCA-secure encryption. We show that in the random oracle model, any public-key adaptively δ-robust PRC can be converted to an $\Omega(\delta)$-robust CCA-secure PRC with a linear information rate.

2.2 Pseudorandom codes

We recall the definition of a public-key PRC.

Definition. *Let Σ be an alphabet and $\mathcal{E} : \Sigma^* \to \Sigma^*$ be an error channel. A* public-key PRC *with oblivious robustness to $\mathcal{E}$ is described by efficient randomized algorithms $\mathsf{Encode}_{\mathsf{pk}} : \Sigma^k \to \Sigma^n$ and $\mathsf{Decode}_{\mathsf{sk}} : \Sigma^* \to \Sigma^k \cup \{\bot\}$, parameterized by keys sk, pk, satisfying the following criteria for every security parameter λ:*

- *(Oblivious robustness) For any message $\mathsf{m} \in \Sigma^k$,*

$$\Pr_{\mathsf{sk},\mathsf{pk}}[\mathsf{Decode}_{\mathsf{sk}}(\mathcal{E}(x)) = \mathsf{m} : x \leftarrow \mathsf{Encode}_{\mathsf{pk}}(\mathsf{m})] \geq 1 - \mathsf{negl}(\lambda)\,.$$

- *(Soundness) For any fixed $c \in \Sigma^*$,*

$$\Pr_{\mathsf{sk}}[\mathsf{Decode}_{\mathsf{sk}}(c) = \bot] \geq 1 - \mathsf{negl}(\lambda)\,.$$

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

$$\left| \Pr_{\mathsf{pk}}[\mathcal{A}^{\mathsf{Encode}_{\mathsf{pk}}}(1^\lambda, \mathsf{pk}) = 1] - \Pr_{\mathsf{pk},\mathcal{U}}[\mathcal{A}^{\mathcal{U}}(1^\lambda, \mathsf{pk}) = 1] \right| \leq \mathsf{negl}(\lambda)\,,$$

  where $\mathcal{A}^{\mathcal{U}}$ means that the adversary has access to an oracle that, on any (even previously queried) input, responds with a freshly drawn uniform value in Σ^n.

If the scheme can only encode a singular message (i.e. $k = 0$), then we call it a *zero-bit* PRC.

Observe that the robustness condition in this definition does not allow the error channel $\mathcal{E}$ to use information about the PRC keys, or even previously seen codewords, to choose the error. Prior results on PRCs only achieve this kind of robustness, which we refer to as *oblivious robustness*.

2.3 Adaptively robust public-key PRCs based on LDPC codes

In this subsection we will outline our proof that the zero-bit LDPC-based PRCs of [CG24] are adaptively robust in the public-key setting. We then show how to build a single-bit adaptively robust PRC, using a different construction from theirs. These results immediately imply the corresponding results in the secret-key setting.

The zero-bit LDPC-based PRC. Let us begin by recalling the zero-bit LDPC-based PRC construction. It will be useful to define the set of all t-sparse vectors in $\mathbb{F}_2^n$,

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

The secret key consists of a collection of $r = n^{\Omega(1)}$ random parity checks $w_1, \ldots, w_r \leftarrow \mathcal{S}_{t,n}$, for some $t = \Theta(\log n)$. Arrange these parity checks into a matrix $\mathsf{sk} = H \in \mathbb{F}_2^{r \times n}$, which will serve as the secret detection key. The public encoding key consists of a random matrix $\mathsf{pk} = G \in \mathbb{F}_2^{n \times d}$ such that $HG = 0$, where $d = \Theta(\log^2 n)$. Let $\mathcal{C}$ be the image of G.

Since this is a zero-bit PRC, we only need to describe a procedure for encoding 1. To encode 1, the encoder outputs $c \oplus e^*$, where $c \leftarrow \mathcal{C}$ and $e^* \leftarrow \mathcal{S}_{\eta n,n}$ for some small constant $\eta > 0$.[6] To decode a string x, the decoder computes $\mathrm{wt}(Hx)$, where wt is the Hamming weight. If $\mathrm{wt}(Hx)$ is significantly less than $r/2$, then the decoder outputs 1; otherwise it outputs $\perp$.

For an appropriate choice of parameters, this scheme is robust to a constant rate of errors *if the errors are chosen independently of H*. In that case, each parity check has an $\Omega(1)^t$ bias towards 0, which is significant if t is small enough. And since there are $r = n^{\Omega(1)}$ independent parity checks in H, a Chernoff bound implies that detection fails with negligible probability. This was essentially the argument used by [CG24] to show that their zero-bit LDPC-based PRCs are robust to a constant rate of errors that are oblivious to the PRC keys H, G.

Of course, this argument breaks down when the errors are allowed to depend on H: Indeed, there is a simple attack that uses H to select $o(n)$ errors that fool the detector.[7] Instead, we will show that this scheme is robust against an adversary that is given G, but not H. Following [CHS24], we refer to this kind of robustness as *adaptive robustness.*

Definition (Definition 3, public-key adaptive δ-robustness, informal)**.** *A public-key pseudorandom code* $(\mathsf{Encode}, \mathsf{Decode})$ *is* adaptively δ-robust *if, for all efficient adversaries $\mathcal{A}$,*

$$\Pr_{\mathsf{sk},\mathsf{pk}}\left[\mathsf{Decode}(\mathsf{sk}, c \oplus e) \neq m \text{ and } \mathrm{wt}(e) \leq \delta n \mid (m, r, e) \leftarrow \mathcal{A}(\mathsf{pk}), c = \mathsf{Encode}(\mathsf{pk}, m; r)\right] \leq \mathsf{negl}(n).$$

In this definition, we allow the adversary to select a message m (which must be $m = 1$ for a zero-bit PRC) and randomness r that together define a codeword $c = \mathsf{Encode}(\mathsf{pk}, m; r)$. The adversary wins if they find a perturbation e, which has weight at most δn, such that $c \oplus e$ does not decode to m.

The secret-key definition is similar, except that the adversary does not get to choose the randomness r and instead interacts with Encode only via oracle access.

Remark. *In this work we only consider robustness to substitution channels. See the paragraph "PRC error model" in Section 1.2 for a discussion on this point.*

Adaptive robustness of the zero-bit LDPC-based PRC. The main technical challenge that we overcome in this work is in proving that the zero-bit LDPC-based PRC is adaptively robust, i.e., that it is robust even when the adversary knows G. More formally, we need to show that any low-weight error vector e that the adversary comes up with must satisfy significantly more than $1/2$ of the parity checks in H. Since we know that there *exist* bad $o(n)$-weight errors that depend on H, our proof strategy must crucially use the fact that e is computed without knowledge of H.

Our first key observation is that, from the perspective of the adversary, H is a uniformly random collection of r vectors chosen from all t-sparse parity checks consistent with G. That is, the rows of H can equivalently be sampled at random from[8]

$$P_{\mathcal{C},t} = \mathcal{C}^\perp \cap \mathcal{S}_{t,n},$$

where $\mathcal{C}$ is the image of G, *at the time of decoding.* This is formalized in Lemma 4.1. Observe that if we choose the dimension of $\mathcal{C}$ as $d = t \log n/2$,

$$|P_{\mathcal{C},t}| \approx \binom{n}{t} \cdot 2^{-d} \approx 2^{t \log n/2},$$

[6] In [CG24], the error was chosen to be i.i.d Bernoulli. For technical reasons, in this work we use random errors of fixed weight instead. In this overview, we also use a star to distinguish the encoding noise e^* from the adversarial perturbation e.

[7] The algorithm uses Gaussian elimination to find an assignment of the first $k = o(n)$ coordinates such that the first k parity checks in H are unsatisfied, if the remaining $n - k$ coordinates are all 0.

[8] We define $\mathcal{C}^\perp = \{w \in \mathbb{F}_2^n : w \cdot c = 0 \ \forall c \in \mathcal{C}\}$.

which is super-polynomial in n since $t = \Theta(\log n)$. Because there are so many parity checks in $P_{\mathcal{C},t}$, we expect that it should be difficult to find low-weight errors that fool $P_{\mathcal{C},t}$.

Indeed, our proof will proceed by showing that for *any* low-weight error e (depending arbitrarily on G), a random parity check from $P_{\mathcal{C},t}$ is satisfied by e with probability significantly greater than $1/2$. The adaptive robustness of our scheme will then follow by a Chernoff bound over the choice of H.

We now turn to our proof that every low-weight error e satisfies significantly more than half of the parity checks in $P_{\mathcal{C},t}$. First, for $S \subseteq \mathbb{F}_2^\ell$ and $z \in \mathbb{F}_2^\ell$, we define

$$\mathsf{bias}(z) = \mathop{\mathbb{E}}_{i \leftarrow [\ell]}[(-1)^{1\{z_i=1\}}] \quad \text{and}$$
$$\mathsf{bias}(S, z) = \mathop{\mathbb{E}}_{w \leftarrow S}[(-1)^{1\{w \cdot z=1\}}].$$

The key ingredient is the following lemma, which serves as the technical backbone of Section 4.

Lemma (Lemma 4.2, informal)**.** *For any code $\mathcal{C} \subseteq \mathbb{F}_2^n$, there is a $\gamma_{\mathcal{C}} \in \mathbb{R}$ such that the following holds. For every $x \in \mathbb{F}_2^n$,*

$$\mathsf{bias}(P_{\mathcal{C},t}, x) = \gamma_{\mathcal{C}} \sum_{c \in \mathcal{C}} \mathsf{bias}(\mathcal{S}_{t,n}, (x \oplus c)).$$

Furthermore,[9] *if $\mathcal{C}$ is a random linear code, then $\gamma_{\mathcal{C}} = 1 \pm \mathsf{negl}(n)$ with probability $1 - \mathsf{negl}(n)$.*

This lemma allows us to reason about the number of parity checks satisfied by x under $P_{\mathcal{C},t}$, by reasoning instead about *random parity checks* from $\mathcal{S}_{t,n}$. See Section 4.3 for the proof of this lemma, which is short, elementary, and motivated by ideas from the analysis of Boolean functions.

If x has low weight, then it turns out that the terms corresponding to non-zero c typically have little contribution in the lemma. This can be viewed as a consequence of the Johnson bound, which says that for a high-distance code there are not many codewords within any given Hamming ball of small radius. Therefore, the lemma implies that[10]

$$\mathsf{bias}(P_{\mathcal{C},t}, e) \approx \mathsf{bias}(\mathcal{S}_{t,n}, e).$$

What we have described so far means that $P_{\mathcal{C},t} = \mathcal{C}^\perp \cap \mathcal{S}_{t,n}$ is enough to essentially capture the structure of all of $\mathcal{S}_{t,n}$, for a random code $\mathcal{C}$. It is easy to see that $\mathsf{bias}(\mathcal{S}_{t,n}, e) \approx \mathsf{bias}(e)^t$, so we can approximate this quantity as a function only of $\mathrm{wt}(e)$ (because $\mathsf{bias}(e) = 1 - 2\,\mathrm{wt}(e)/n$).

The final step is to recall Lemma 4.1, which says that we can sample the rows of H as a random subset of r parities from $P_{\mathcal{C},t}$ *after* the adversary has decided on e. By a Chernoff bound over the choice of H, it follows that $\mathsf{bias}(He) \approx \mathsf{bias}(P_{\mathcal{C},t}, e)$. Together with the approximations $\mathsf{bias}(P_{\mathcal{C},t}, e) \approx \mathsf{bias}(\mathcal{S}_{t,n}, e)$ and $\mathsf{bias}(\mathcal{S}_{t,n}, e) \approx \mathsf{bias}(e)^t$, we finally have that

$$\mathsf{bias}(He) \approx \mathsf{bias}(e)^t. \tag{1}$$

With appropriate choice of parameters, this completes the proof of zero-bit adaptive robustness.

Adaptive robustness of single-bit LDPC-based PRCs. We now describe a single-bit PRC that is adaptively δ-robust for any $\delta < 1/4$. The reader might wonder why a zero-bit PRC cannot be immediately used as a single-bit PRC by re-interpreting $\perp$ as 0. But with this re-interpretation, zero-bit robustness only requires that it is hard for an adversary to turn an encoding of 1 into an encoding of 0, while single-bit robustness *also* requires that the adversary cannot turn an encoding of 0 into an encoding of 1.

Instead, our strategy is to generate two independent pairs of zero-bit adaptively δ-robust PRC keys (H_0, G_0) and (H_1, G_1), and to define the encoding of $m \in \{0, 1\}$ as a noisy sample from the image of G_m.[11] The

[9] Technically, this part is Fact 4.2.

[10] This approximation is not strictly true in general, but we prove it for the relevant regimes of e.

[11] Actually we only allow the encoder to output *non-zero* vectors. This is because the adversary in the public-key adaptive robustness game chooses the encoding randomness, and the zero vector would trivially violate robustness.

decoder outputs $m \in \{0,1\}$ if the given string decodes to 1 under the H_m zero-bit decoder *and* it decodes to $\perp$ under the H_{1-m} zero-bit decoder. If both zero-bit decoders output 1 or $\perp$, then our single-bit decoder cannot determine which bit is encoded, so it outputs $\perp$.

Adaptive δ-robustness of the zero-bit PRC implies that the adversary cannot find a δn-weight error that causes both zero-bit decoders to output $\perp$. However, it says nothing about the possibility of causing both zero-bit decoders to output 1. In other words, we still need to rule out the possibility that the adversary can sample a codeword under G_0 or G_1 and produce a nearby string that decodes to 1 under *both* H_0 and H_1.

Let $\mathcal{C}_m$ be the image of G_m for $m \in \{0,1\}$. The idea is to use the fact that $\mathcal{C}_0$ and $\mathcal{C}_1$ are well-separated with high probability: That is, the closest non-zero pair of codewords $c_0 \in \mathcal{C}_0, c_1 \in \mathcal{C}_1$ are approximately $n/2$-far with probability $1 - \mathsf{negl}(n)$ over G_0, G_1. Therefore, if the adversary produces $c \oplus e$ where $c \in \mathcal{C}_m$ and $\text{wt}(e) \leq \delta n$, then $c \oplus e$ will be at least roughly $(1/2 - \delta)n$-far from any codeword in $\mathcal{C}_{1-m}$. For $\delta < 1/4$, this yields our separation: $c \oplus e$ is $\delta n < n/4$ far from $\mathcal{C}_m$, and $(1/2 - \delta)n > n/4$ far from $\mathcal{C}_{1-m}$.

So the question is whether our parity check matrices H_0, H_1 will be able to observe this difference. But fortunately, this was already addressed in Equation (1)! The only thing that remains is to account for parameters, making sure that the zero-bit decoder thresholds are set appropriately so as not to detect beyond the δn radius desired.

Remark. *While we construct zero-bit PRCs that are adaptively δ-robust for any $\delta < 1/2$, it is not possible to construct single-bit PRCs for $\delta \geq 1/4$, as demonstrated by the following attack. The attacker draws random x_0 and x_1, encoding 0 and 1 respectively. By pseudorandomness, x_0 and x_1 must be equal on roughly half of their locations. Therefore, the adversary can craft x' that differs from each of x_0 and x_1 on at most a $1/4$ fraction of locations. Since x' cannot decode to both 0 and 1, robustness is violated for either x_0 or x_1.*

2.4 Ideal PRCs: the secret-key setting

So far, we've considered robustness against an adversary with access to only the encoder. In this section, we give the adversary access to both encoding *and* decoding oracles.[12] An ideal PRC should retain both robustness and pseudorandomness in this setting.

Defining pseudorandomness requires some care in this case. Of course, an adversary can distinguish a codeword from random simply by submitting the codeword as a decoding query. Nonetheless, we find that robustness and pseudorandomness can be elegantly combined into a single definition in the secret-key setting, which we call *ideal security.* This definition uses the real/ideal world paradigm, common in cryptography. Ideal security requires that no adversary can distinguish between the ideal world, where the challenger responds to encoding and decoding queries by comparing to previous responses; and the real world, where the challenger responds according to the PRC algorithms.

The ideal world. A PRC simply produces codewords that are pseudorandom, which can be decrypted even when subjected to errors. Strings that are far from all observed codewords should decode to $\perp$. Given this, an ideal δ-robust PRC should satisfy the following requirements:

- Responses to encoding queries should appear random.
- Any string that is within δ of an observed codeword should decode to its underlying message.
- Any string that is at least δ-far from all observed codewords should decode to $\perp$.

These requirements *fully specify* the behavior of the PRC in the ideal world. Therefore, we define the ideal world as follows. The challenger responds to each encoding query with a fresh uniformly random string, storing the queried message and response in memory. The challenger responds to each decoding query by

[12] Since the adversary is only given to encoding and decoding oracles, this is the *secret-key* setting. In the next section we will consider adversaries with access to an encoding key and a decoding oracle.

checking if the given string is close to any response in memory. If so, it returns the corresponding message; otherwise, it returns $\perp$.

Ideal security. A PRC satisfies ideal security if no adversary can distinguish between the real world and the ideal world. Observe that ideal security constitutes a complete definition of a PRC: Whereas an adaptively robust PRC needs to satisfy separate definitions of soundness, pseudorandomness, and adaptive robustness, ideal security encompasses all of these properties at once.

Proving ideal security. We show that the existence of an adaptively δ-robust single-bit PRC implies the existence of an ideal δ-robust PRC with linear information rate.

We first show that the single-to-multi-bit PRC transformation from [CG24] preserves adaptive robustness. This transformation is simple. Let PRC be any single-bit adaptively robust secret-key PRC, let PRG be a pseudorandom generator, and let ECC be an error-correcting code. An encoding of m takes the form

$$\pi(\mathsf{PRC.Encode}_{\mathsf{sk}}(r_1)||\dots||\mathsf{PRC.Encode}_{\mathsf{sk}}(r_\lambda), \mathsf{PRG}(r) \oplus \mathsf{ECC}(m)), \tag{2}$$

where $r \leftarrow \{0,1\}^\lambda$ is a fresh sample for each encoding and π is a random permutation of the bits which is included as part of the key. The decoder is straightforward. The random permutation ensures that any errors introduced by the adversary are balanced across the blocks encoding the r_i's and the error-corrected message.[13] Note also that if ECC has linear rate, then so does this resulting PRC (since λ does not need to grow with the message length).

We now turn to showing that one can use any adaptively robust PRC with polynomial information rate, PRC, to construct one satisfying ideal security. The basic strategy is to build a scheme that satisfies ideal security in the case that the adversary makes only one decoding query. Once this is established, complete ideal security will follow directly from a hybrid argument.

The main challenge is that, perhaps unintuitively, the PRC may sometimes be *too* robust. Ideal δ-robustness requires a *sharp* decoding threshold: any string that is δ-far from all observed codewords must decode to $\perp$. If our decoder accepts strings outside of this threshold, then it could accidentally leak information about the key! Therefore, we need our decoder to be able to determine *precisely* if a given string is within distance δn of a previously seen codeword.

Suppose an adversary takes a codeword c and adds some (possibly greater than δn-weight) error to obtain c'. Our key idea is that if the decoder can somehow recover from c' both the message and randomness used to generate c, then it can recover c and exactly compute its distance from c'. We define PRC^{sharp} to allow exactly this, by drawing a random $r \leftarrow \{0,1\}^\lambda$ and letting the encoding of a message m under PRC^{sharp} be[14]

$$c \leftarrow \mathsf{PRC.Encode}_{\mathsf{sk}}(r||\mathsf{m}; \mathsf{PRF}_{\mathsf{sk}}(r)),$$

where PRF is a pseudorandom function. In words, we encode a seed r as part of the message, and use $\mathsf{PRF}_{\mathsf{sk}}(r)$ as the randomness in the encoding. Pseudorandomness of PRF implies that codewords of PRC^{sharp} are indistinguishable from those of PRC. Now suppose we are given c' that is δ-far from c yet still decodes validly under $\mathsf{PRC.Decode}_{\mathsf{sk}}$. Our PRC^{sharp} decoder will then recover m and r, recompute c, and observe that c' is too far from c. It then knows to output $\perp$.

We've now addressed the issue where c' is too far from a previously seen c, but still decodes validly to the message encoded by c. But there is still the possibility that the adversary produces a c' that decodes to

[13] This point is actually quite subtle. Because it is not possible to test whether an error is successful without the key, we cannot rely directly on pseudorandomness here. Instead, we use the fact that it *is* possible to test whether the errors are balanced using only π, but not the underlying PRC key.

[14] Note that this is essentially the Fujisaki-Okamoto transformation [FO99], but in a secret-key setting where a PRF suffices instead of a random oracle.

some message that it has never queried. We address this issue by simply adding an authentication tag to the message. That is, our final PRC^{sharp} codewords take the form

$$c \leftarrow \mathsf{PRC.Encode}_{\mathsf{sk}}(r||\mathsf{m}||R_1; R_2),$$

where $(R_1, R_2) = \mathsf{PRF}_{\mathsf{sk}}(r||\mathsf{m})$. Now, if a received string decodes to $r||\mathsf{m}||R_1$ under $\mathsf{PRC.Decode}_{\mathsf{sk}}$, the PRC^{sharp} decoder checks that $R_1 = \mathsf{PRF}_{\mathsf{sk}}(r||\mathsf{m})$. If so, it outputs m; otherwise it outputs $\perp$. Pseudorandomness of PRF implies that the adversary cannot produce an accepting r, m, R_1 that were not included in a previous response.

Therefore, if the adversary produces x that decodes to anything other than $\perp$, it must be the case that x is δ-far from some previously-seen codeword produced by the challenger. This yields single-decoding-query ideal security, from which ideal security follows by a simple hybrid argument.

Remark. *Multi-bit δ-ideal security is only possible for $\delta < 1/4$, by the same example in the remark at the end of Section 2.3. However, we proved that adaptive robustness is possible up to $1/2$ for* zero-bit *PRCs. It would be interesting to construct zero-bit PRCs that satisfy δ-ideal security for $\delta \geq 1/4$, but it appears that new ideas would be needed.*

2.5 CCA security: the public-key setting

In this subsection we will outline our definition and construction of CCA-secure PRCs.

Single-bit to multi-bit PRC transformation. We first present a simple transformation from an adaptively robust single-bit PRC to an adaptively robust PRC with linear rate, in the public-key setting. In the public-key setting, the transformation described earlier in Equation (2) is no longer robust because it relies on the secrecy of the permutation π.

Our solution is to first encode r in an error-correcting code, and encode each bit of the resulting codeword separately under the single-bit PRC. We also make the block encoding r and the block encoding the message equal lengths. Now, an adversary introducing a δ rate of errors can introduce at most a 2δ rate of errors to either the message block or the r block. If the errors on the r block are concentrated on any given single-bit PRC, then the error-correcting code handles them; if they are spread out among the single-bit PRCs, then the PRC robustness handles them. This transformation is quite lossy in terms of robustness, which is at most $1/32$ regardless of the underlying PRC and error-correcting codes. The information rate is also at most half that of the error-correcting code. It would be interesting to come up with a better transformation, but we do not pursue this here. In any case, we still achieve $\Omega(1)$-robustness and a linear information rate.

CCA security: definition. Our CCA (chosen codeword attack) security definition subsumes both pseudorandomness and adaptive robustness. It also generalizes CCA security for pseudorandom encryption. Roughly speaking, the security definition states that pseudorandomness of public-key PRC should be secure against computationally bounded adversaries even when given the public key and access to the decoding oracle. This definition will be modeled similar to the private-key setting, except that we need to additionally take into consideration that the adversary can create its own codewords. We formalize the security definition by again considering a real and a "random" experiment. Whereas we refer to the latter as the "ideal" world in the secret-key setting, in the public-key setting we use a different word because it is not clearly "ideal." In particular, in the public-key setting our definition of the random world *still makes use of the real PRC decoder.*

The random world. In the random game, the adversary receives the following:

- The public encoding key pk,
- Access to the random encoding oracle: the random encoding oracle on input a message m, outputs a random string c as a codeword.

- Access to the corresponding decoding oracle: upon receiving a codeword c', this oracle first checks if c' is close to any of the codewords returned by the random encoding oracle. If so, it returns m, where c was returned as a response to the encoding query m. If c' is not close to any of the outputs of the encoding oracle then it responds according to the actual decoder on input c'.

The real world. In the real game, the adversary receives the public encoding key pk and has access to the real encoding and decoding oracles.

CCA security simply says that any computationally bounded adversary cannot distinguish whether it is participating in the random or the real game.

Achieving CCA security generically. We show how to achieve CCA security generically starting with any adaptively robust PRC with polynomial information rate in the random oracle model. Our transformation is essentially identical to the ideal PRC transformation, except that we use a random oracle instead of a PRF. This makes it very similar to the Fujisaki-Okamaoto (FO) transformation [FO99], although the analysis is slightly different. Specifically, we crucially rely upon the adaptive robustness property in the security proof unlike the FO transformation.

Suppose PRC is any adaptively robust pseudorandom code supporting multi-bit messages. We construct a CCA-secure pseudorandom code $\mathsf{PRC}^{\mathsf{CCA}}$ as follows:

- The key generation algorithm of $\mathsf{PRC}^{\mathsf{CCA}}$ simply runs the key generation of the PRC to generate $(\mathsf{pk}, \mathsf{sk})$. It also samples F from a hash function family; F will be modeled as a random oracle in the security proof. The new public key of $\mathsf{PRC}^{\mathsf{CCA}}$ is set to be (pk, F) and the new secret key is set to be sk.
- The encode algorithm of $\mathsf{PRC}^{\mathsf{CCA}}$, on input a message m, does the following. It first samples a λ-bit string r uniformly at random. It then computes $F(m, r)$ to obtain a string that can be broken down into two parts (R_1, R_2), each of length at least λ. It then encodes the new message (m, r, R_2) using the encode algorithm of PRC and using the randomness R_1. Denote the resulting codeword to be c.
- The decode algorithm, on input the secret key sk and the codeword c, does the following: it first runs the PRC decoding algorithm on c to obtain (m, r, R_2'). It then computes $F(m, r)$ to obtain (R_1, R_2). It outputs m if $R_2 = R_2'$. Otherwise, it outputs $\perp$.

To prove the CCA security of the above scheme, we undertake the following three steps.

In the **first step**, we observe the queries made by the adversary. Specifically, we consider the following event.

> $\underline{\mathsf{NeverQueried}}$: The adversary, during the decoding query phase, submits a codeword c that decodes to (m, r, R_2) such that (m, r) has never been queried by the adversary to F.

We argue that $\mathsf{NeverQueried}$ only happens with negligible probability. Indeed, if the adversary has never queried F on (m, r) then the probability that it predicts R_2 is negligible in λ.

This suggests that there are only two types of decoding queries c that the adversary can submit. Either

- c is close to one of the codewords returned by the challenger to the adversary during the encoding phase, or
- c decodes to (m, r, R_2), where (m, r) is a random oracle query made by the adversary.

This suggests that the challenger does not need the decoding key to answer the decoding queries at all! It can answer just by (a) observing the adversarial queries to the F and, (b) by keeping tracking of the codewords it returns during the encoding phase.

This leads us to the **second step**. In the second step, the challenger does not use the decoding key to answer the decoding queries. Instead it uses an alternate decoding procedure, referred to as $\mathsf{AltDecode}$. Upon receiving a codeword c during the decoding query phase, $\mathsf{AltDecode}$ first performs the following checks:

- It checks if c is close to any of the codewords returned during the encoding phase,
- It checks if c is to close to any of the codewords created by the adversary using the queries to F. In more detail, for every adversarial query (m, r), create a codeword with (m, r, R_2) as the message and R_1 as the randomness, where (R_1, R_2) is the output of F on (m, r). Check if c is close to any of the created codewords.

If any of these two checks pass,[15] return the message encoded in the codeword close to c. Else, return $\perp$.

To argue that $\mathsf{AltDecode}$ simulates the use of the real decoding algorithm during the decoding phase, we need to argue that for every decoding query, the output of $\mathsf{AltDecode}$ and the real decoder is the same. If not, then we claim that the adaptive robustness property is violated. This is because the adversary efficiently came up with two different, but close, codewords that open to two different messages. Thus, from the adaptive robustness property, we can conclude that the outputs of $\mathsf{AltDecode}$ and the real decoder are the same for every decoding query.

In the **third step**, we invoke the pseudorandomness guarantee of PRC to switch all the codewords generated during the encoding phase to be uniformly random strings. In order to do this switch, it was crucial that the challenger was using $\mathsf{AltDecode}$, which in turn does not use any secret key, in the decoding phase.

All the three steps combined prove the CCA security of $\mathsf{PRC}^{\mathsf{CCA}}$.

3 Preliminaries

For a randomized algorithm $A(\cdot)$, we write $A(x; r)$ to denote the output of A on input x and randomness r.

For a set X, we define $X^* = \{(x_1, \ldots, x_k) \mid x_1, \ldots, x_k \in X \wedge k \in \mathbb{Z}_{\geq 0}\}$ to be the set of all strings over the alphabet X. For a binary string $s \in X^*$, we let s_i denote the i^{th} symbol of s and $\operatorname{len} s$ denote the length of s.

We write $[n] = \{1, \ldots, n\}$. Let $x \in \{0,1\}^n$ and let π be a permutation over $[n]$. We let $\mathsf{PermBits}(x, \pi)$ denote the function that outputs $x_{\pi(1)} || \cdots || x_{\pi(n)} \in \{0,1\}^n$.

Lemma 3.1 (Hypergeometric tail bounds [Hoe94])**.** *Let $X \sim \mathrm{Hyp}(N, K, n)$ and $p = K/N$. Then for any $0 < t < K/N$,*

$$\Pr\left[X \leq (p - t)n\right] \leq e^{-2t^2 n}, \text{ and}$$
$$\Pr\left[X \geq (p + t)n\right] \leq e^{-2t^2 n}.$$

Lemma 3.2 (Chernoff bounds)**.** *Let $X_1, \ldots, X_n \in [0,1]$ be independent random variables. Let $\mu = \mathbb{E}\left[\sum_{i=1}^n X_i\right]$. Then for any $\delta \in (0,1)$:*

$$\Pr\left[\sum_{i=1}^n X_i \geq (1+\delta)\mu\right] \leq \exp\left(-\frac{\mu\delta^2}{3}\right) \text{ and}$$
$$\Pr\left[\sum_{i=1}^n X_i \leq (1-\delta)\mu\right] \leq \exp\left(-\frac{\mu\delta^2}{2}\right).$$

[15] It could be that there is more than one codeword that is close to c. In this case, pick one of them at random.

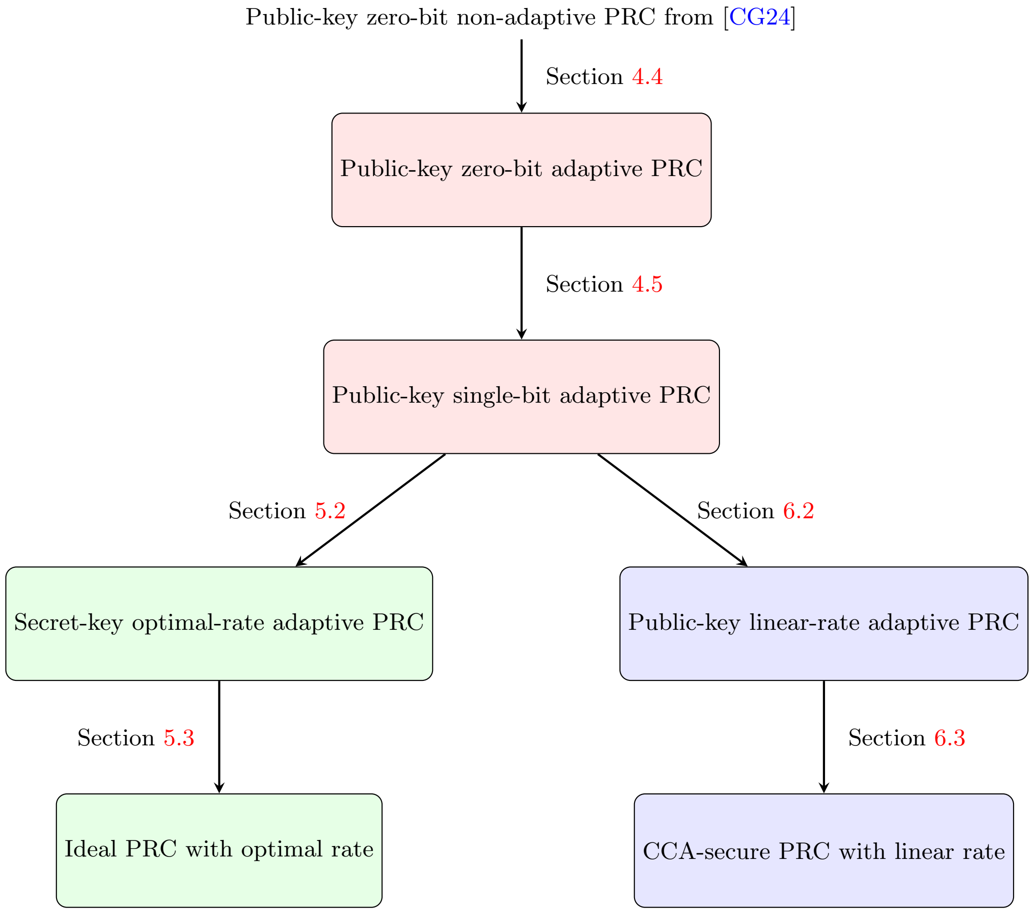
Public-key zero-bit non-adaptive PRC from [CG24]
Section 4.4
Public-key zero-bit adaptive PRC
Section 4.5
Public-key single-bit adaptive PRC
Section 5.2
Section 6.2
Secret-key optimal-rate adaptive PRC
Public-key linear-rate adaptive PRC
Section 5.3
Section 6.3
Ideal PRC with optimal rate
CCA-secure PRC with linear rate

Figure 1: Organization of the paper. The red boxes are particular to the LDPC-based public-key PRCs related to [CG24], the green boxes are generic in the secret-key setting, and the purple boxes are generic in the public-key setting.

Coding theory notation. Let n be the dimension, let t be an even integer, and let $d = \Theta(\log \binom{n}{t})$. Let $W_t \in \mathbb{F}_2^{\binom{n}{t} \times n}$ be the matrix whose rows are all weight-t parity checks. For a linear code $\mathcal{C} \subseteq \mathbb{F}_2^n$, let $P_{\mathcal{C},t}$ be the matrix whose rows are all weight-t parity checks satisfied by $\mathcal{C}$. Let $N_{\mathcal{C},t}$ be the number of rows in $P_{\mathcal{C},t}$. For $z \in \{0,1\}^m$, we define $\mathsf{bias}(z) = \frac{1}{m}\sum_{i=1}^m (-1)^{z_i}$. Note that $\mathsf{bias}(z) = 1 - 2\,\mathrm{wt}(z)/n$ and $\mathrm{wt}(z) = (1/2 - \mathsf{bias}(z)/2) \cdot n$.

Cryptography preliminaries. A pseudorandom function is a function that behaves indistinguishably from a random function, from the perspective of any computationally-bounded adversary that makes only black-box queries to the function. Pseudorandom functions are equivalent to one-way functions [GGM86], the minimal object of classical cryptography.

Pseudorandom function (PRF). Let $\mathcal{F} = \{F_{\mathsf{sk}} : \{0,1\}^{\ell_1(\lambda)} \to \{0,1\}^{\ell_2(\lambda)} \mid \mathsf{sk} \in \{0,1\}^\lambda\}$ be a family of functions. $\mathcal{F}$ is a PRF if F_{sk} is efficiently computable and for all polynomial-time distinguishers D,

$$\left| \Pr_{\mathsf{sk} \leftarrow \{0,1\}^\lambda} \left[D^{F_{\mathsf{sk}}(\cdot)}(1^\lambda) = 1 \right] - \Pr_{f} \left[D^{f(\cdot)}(1^\lambda) = 1 \right] \right| \leq \mathsf{negl}(\lambda) \,.$$

where f denotes a random function from $\{0,1\}^{\ell_1(\lambda)}$ to $\{0,1\}^{\ell_2(\lambda)}$.

3.1 Pseudorandom codes

We recall the definition of a public-key pseudorandom code (PRC) with oblivious robustness from [CG24]. We present only the public-key definition; the secret-key version is identical except that the public key is included in the secret key, and the adversary in the pseudorandomness condition is only given 1^λ as input. Secret-key PRCs are defined fully in [CG24].

Definition 1 (Public-key PRC)**.** *Let Σ be a fixed alphabet. A* public-key pseudorandom error-correcting code *(abbreviated as public-key PRC) with (oblivious) robustness to a channel $\mathcal{E} : \Sigma^* \to \Sigma^*$ is a triple of polynomial-time randomized algorithms* $(\mathsf{KeyGen}, \mathsf{Encode}, \mathsf{Decode})$ *satisfying*

- *(Syntax) There exist functions $\ell_{\mathsf{Dec}}, \ell_{\mathsf{Enc}}, n, k : \mathbb{N} \to \mathbb{N}$ such that for all $\lambda \in \mathbb{N}$,* $\mathsf{KeyGen}(1^\lambda) \in \{0,1\}^{\ell_{\mathsf{Dec}}(\lambda)} \times \{0,1\}^{\ell_{\mathsf{Enc}}(\lambda)}$, $\mathsf{Encode}(1^\lambda, \mathsf{pk}, \mathsf{m}) \in \Sigma^{n(\lambda)}$ *takes inputs* $\mathsf{pk} \in \{0,1\}^{\ell_{\mathsf{Enc}}(\lambda)}$, $\mathsf{m} \in \Sigma^{k(\lambda)}$, *and* $\mathsf{Decode}(1^\lambda, \mathsf{sk}, x) \in \Sigma^{k(\lambda)} \cup \{\bot\}$ *takes inputs* $\mathsf{sk} \in \{0,1\}^{\ell_{\mathsf{Dec}}(\lambda)}$, $x \in \Sigma^*$.
- *(Oblivious robustness) For any $\lambda \in \mathbb{N}$ and any message $\mathsf{m} \in \Sigma^{k(\lambda)}$,*

$$\Pr_{(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)}[\mathsf{Decode}(1^\lambda, \mathsf{sk}, \mathcal{E}(x)) = \mathsf{m} : x \leftarrow \mathsf{Encode}(1^\lambda, \mathsf{pk}, \mathsf{m})] \geq 1 - \mathsf{negl}(\lambda)\,.$$

- *(Soundness) For any fixed $c \in \Sigma^*$,*

$$\Pr_{(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)}[\mathsf{Decode}(1^\lambda, \mathsf{sk}, c) = \bot] \geq 1 - \mathsf{negl}(\lambda)\,.$$

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

$$\left| \Pr_{(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)}[\mathcal{A}^{\mathsf{Encode}(1^\lambda, \mathsf{pk}, \cdot)}(1^\lambda, \mathsf{pk}) = 1] - \Pr_{\substack{(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \mathcal{U}}}[\mathcal{A}^{\mathcal{U}}(1^\lambda, \mathsf{pk}) = 1] \right| \leq \mathsf{negl}(\lambda)\,,$$

  where $\mathcal{A}^{\mathcal{U}}$ means that the adversary has access to an oracle that, on any (even previously queried) input, responds with a freshly drawn uniform value in $\Sigma^{n(\lambda)}$.

We say that a PRC is "zero-bit" if it can encode only a singular message ($\Sigma = \{1\}$ and $k = 0$), or "single-bit" if it can encode two messages ($\Sigma = \{0,1\}$ and $k = 1$).

For completeness, we include the original oblivious robustness definition from [CG24]. However, in this work we focus on stronger notions of robustness, which in particular imply oblivious robustness to substitution channels.

4 Adaptively robust public-key PRCs based on LDPC codes

This section is dedicated to proving that variants of (public-key) PRCs from [CG24] satisfy the adaptive robustness notion (without decoder access) we introduce in this work. Correspondingly, in Section 4.1 we give our new notion of adaptive robustness for both the public-key and the secret-key PRC variants (where the former is clearly stronger than the latter). In Section 4.2 we review the *zero-bit* (public-key) PRC from [CG24], in Section 4.3 we prove some technical tools that we will use, in Section 4.4 we prove the (public-key, and hence, also secret-key) adaptive security of this construction. And, finally, in Section 4.5 we show how to extend the resulting zero-bit construction to a single-bit adaptively-secure (still public-key) PRC.

4.1 Definitions

We define adaptive robustness in the secret-key setting via the following security game.

$\underline{\mathcal{G}^{\mathsf{robust-sk}}_{\mathcal{A},\mathsf{PRC},\delta}(1^\lambda)\text{:}}$

1. The challenger sets $\mathsf{transcript} = \emptyset$ and samples $\mathsf{sk} \leftarrow \mathsf{PRC.KeyGen}(1^\lambda)$.
2. The adversary is allowed to make encoding queries. For each encoding query m, the challenger responds with $c \leftarrow \mathsf{PRC.Encode}(\mathsf{sk}, m)$ and sets $\mathsf{transcript} = \mathsf{transcript} \cup \{(m, c)\}$.
3. The adversary sends the challenger x.
4. The challenger computes $m' = \mathsf{PRC.Decode}(\mathsf{sk}; x)$ (which could be $\bot$).
5. If there exists $(m, c) \in \mathsf{transcript}$ such that $\mathrm{wt}(x \oplus c) \leq \delta n$ and $m \neq m'$, then the adversary wins; otherwise the adversary loses.

Definition 2 (Secret-key adaptive robustness)**.** *We say that a secret-key pseudorandom code* PRC *is* adaptively δ-robust *if, for any efficient adversary $\mathcal{A}$,*

$$\Pr\left[\mathcal{A} \textit{ wins } \mathcal{G}^{\mathsf{robust-sk}}_{\mathcal{A},\mathsf{PRC},\delta}(1^\lambda)\right] \leq \mathsf{negl}(\lambda)\,.$$

The public-key setting is similar, except that the adversary gets the encoding key itself rather than merely oracle access to the encoder. Since the adversary can therefore encode messages on their own, we require the adversary to submit a message and randomness that witness the failure of the decoder.

$\underline{\mathcal{G}^{\mathsf{robust-pk}}_{\mathcal{A},\mathsf{PRC},\delta}(1^\lambda)}$:

1. The challenger samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PRC.KeyGen}(1^\lambda)$ and sends pk to the adversary.
2. The adversary sends (m, r, x) to the challenger.
3. The challenger computes $c = \mathsf{PRC.Encode}(\mathsf{pk}, m; r)$ and $m' = \mathsf{PRC.Decode}(\mathsf{sk}, x)$. If $\mathrm{wt}(x \oplus c) \leq \delta n$ and $m' \neq m$, then the adversary wins; otherwise the adversary loses.

Definition 3 (Public-key adaptive robustness)**.** *We say that a public-key pseudorandom code* PRC *is* adaptively δ-robust *if, for any efficient adversary $\mathcal{A}$,*

$$\Pr\left[\mathcal{A} \textit{ wins } \mathcal{G}^{\mathsf{robust-pk}}_{\mathcal{A},\mathsf{PRC},\delta}(1^\lambda)\right] \leq \mathsf{negl}(\lambda)\,.$$

Note that Definition 3 is a stronger definition than Definition 2 — any scheme satisfying Definition 3 automatically satisfies Definition 2. We therefore devote this section to proving public-key adaptive robustness of the PRC from [CG24], which immediately implies the secret-key adaptive robustness of the same PRC.

At this point, a few remarks are in order about our choice of definitions.

Remark. *The reader may wonder why we don't just have the adversary produce two nearby strings x, x' that decode to different values. This would be an impossibly strong definition: The adversary could choose x with slightly fewer errors than the scheme tolerates, and x' with slightly more. A successful adversary must come up with a* valid *codeword, together with a low-weight modification of it that decodes incorrectly.*

Remark. *It is possible to define public-key robustness along the same lines as Definition 2 (secret-key adaptive robustness). That is, we could consider the following game: the adversary has adaptive access to the encoding oracle and later on, is expected to come up with a string c that is close to one of the codewords, say c', returned by the encoding oracle. It wins if c' is an encoding of m and c does not decode to m. There are a couple of reasons behind our choice of the adaptive robustness definition in the public-key setting (Definition 3). First, Definition 3 is stronger than the public-key analogue of Definition 2, and we will see that our scheme satisfies this stronger notion anyways. Second, the definition is more compact and easier to work with. In particular, we crucially invoke this stronger Definition 3 in our later proofs about CCA security (see the proof of Lemma 6.3).*

Remark. *One can also similarly define a stronger version of our symmetric-key Definition 2, where the attacker can also control the randomness for the encoding oracle (and not only the message). Of course, our scheme will satisfy this stronger definition too (as it satisfies the public-key analog of this strengthening given in Definition 3). However, we did not choose to follow this route for several reasons. First, unlike the public-key case, this definition is not significantly more compact or intuitive than our definition. Second, we do not have any real-world motivation for this notion, and unlike the public-key case, we do not require the stronger notion for any proofs later on in the paper. Third, unlike the public-key setting, there is a simple generic transformation from our Definition 2 to the stronger variant: instead of sampling randomness r for* PRC.Encode *directly, we sample auxiliary randomness s, and set $r = \mathsf{PRF}_{\mathsf{sk}}(m, s)$, where $\mathsf{PRF}_{\mathsf{sk}}$ is a pseudorandom function whose key is part of the overall secret key.*

4.2 The scheme

Here we recall the LDPC-based zero-bit PRC construction from [CG24]. For technical reasons, we modify the definition slightly to use fixed-weight error instead of Bernoulli.

Let

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

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

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

be the set of all t-row-sparse matrices in $\mathbb{F}_2^{r \times n}$.

Our zero-bit pseudorandom LDPC codes are parameterized by a public generator matrix $G \in \mathbb{F}_2^{n \times d}$ and a secret parity-check matrix $H \in \mathbb{F}_2^{r \times n}$. The sampling process for these matrices is described in Definition 4.

Definition 4 (Random LDPC code, $\mathsf{LDPC}[n, d, t, r]$)**.** *For $n, d, t, r \in \mathbb{N}$, define the distribution $\mathsf{LDPC}[n, d, t, r]$ over $\mathbb{F}_2^{r \times n} \times \mathbb{F}_2^{n \times d}$ as follows:*

$\mathsf{LDPC}[n, d, t, r]$*:*

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).*

An (n, d, t, r) random LDPC code *is a pair of matrices $(H, G) \leftarrow \mathsf{LDPC}[n, d, t, r]$.*

We now define our LDPC-based zero-bit PRC. Recall that a zero-bit PRC is one whose message space is just $\{1\}$. The following construction differs slightly from that in [CG24], in that the error distribution is uniform over $S_{\eta n,n}$ instead of $\mathrm{Ber}(n, \eta)$.

Construction 1 (Zero-bit public-key pseudorandom LDPC code, $\mathsf{LDPC\text{-}PRC}_0[n, d, t, r, \eta, \zeta]$)**.** *Let $n, d, t, r : \mathbb{N} \to \mathbb{N}$ and $\eta, \zeta : \mathbb{N} \to [0, 1/2)$ be efficiently-computable functions of the security parameter. We define $\mathsf{LDPC\text{-}PRC}_0[n, d, t, r, \eta, \zeta]$ by the following algorithms, where we leave the dependence of n, d, t, r, η, ζ on λ implicit:*

- $\mathsf{KeyGen}(1^\lambda)$*: Sample $(H, G) \leftarrow \mathsf{LDPC}[n, d, t, r]$ and $z \leftarrow \mathbb{F}_2^n$. Output $(\mathsf{sk} = (H, z), \mathsf{pk} = (G, z))$.*
- $\mathsf{Encode}(1^\lambda, (G, z))$*: Sample $u \leftarrow \mathbb{F}_2^d$, $e \leftarrow \mathcal{S}_{\eta n,n}$. Output $Gu \oplus z \oplus e$.*
- $\mathsf{Decode}(1^\lambda, (H, z), x)$*: If $\mathrm{wt}(H(x \oplus z)) < \left(\frac{1}{2} - \zeta\right) \cdot r$, output 1; otherwise output $\perp$.*

The original construction of [CG24] is pseudorandom under the subexponential LPN assumption. Since ours is the same except that we used fixed-weight error, ours is pseudorandom under the polynomially related *exact LPN* assumption:

Assumption 1 (Exact LPN assumption [JKPT12])**.** *For $\eta \in (0, 1/2)$ and $g : \mathbb{N} \to \mathbb{N}$, the $\mathsf{xLPN}_{g,\eta}$ assumption states that for every $n \in \mathbb{N}$ and every polynomial-time adversary $\mathcal{A}$,*

$$\left| \Pr_{\substack{A \leftarrow \mathbb{F}_2^{n \times g(n)} \\ s \leftarrow \mathbb{F}_2^{g(n)} \\ e \leftarrow \mathcal{S}_{\eta n, n}}} [\mathcal{A}(A, As \oplus e) = 1] - \Pr_{\substack{A \leftarrow \mathbb{F}_2^{n \times g(n)} \\ u \leftarrow \mathbb{F}_2^n}} [\mathcal{A}(A, u) = 1] \right| \leq \mathsf{negl}(n)\,.$$

Similarly to standard LPN, the exact LPN assumption implies that any polynomial number of samples of the form $(A, As \oplus e)$ are indistinguishable from uniformly random samples.

Importantly, the exact LPN assumption is *equivalent* to the standard LPN assumption.

Fact 4.1 (Proposition 2.3 from [JKPT12])**.** *The hardness of $\mathsf{xLPN}_{g,\eta}$ is polynomially related to the hardness of $\mathsf{LPN}_{g,\eta}$ as defined in [CG24].*

4.3 The toolkit

The basic idea is to use the fact that there are actually $n^{\Omega(\log n)}$ parity checks that are satisfied by a random $\mathcal{C}$, even though our detector uses just $r = O(n)$ of them. Since the adversary only sees $\mathcal{C}$, from their perspective the actual parity checks used by our detector are chosen uniformly at random from $P_{\mathcal{C},t}$. This is formalized in Lemma 4.1.

Lemma 4.1. *Let $H \in \mathbb{F}_2^{r \times n}$ be a random matrix where each row is t-sparse. Let $\mathcal{C}$ be a random d-dimensional subspace selected from $\ker H$. Then for all d-dimensional subspaces $\mathcal{C}_* \subseteq \mathbb{F}_2^n$ and all $H_*, H'_* \in \mathbb{F}_2^{r \times n}$ such that $\mathcal{C}_* \subseteq \ker H_* \cap \ker H'_*$,*

$$\Pr[H = H_* \mid \mathcal{C} = \mathcal{C}_*] = \Pr[H = H'_* \mid \mathcal{C} = \mathcal{C}_*].$$

Proof. Since $\mathcal{C}$ is selected uniformly at random from all d-dimensional subspaces of $\ker H$,

$$\Pr[\mathcal{C} = \mathcal{C}_* \mid H = H_*] = \Pr[\mathcal{C} = \mathcal{C}_* \mid H = H'_*].$$

Since H is selected uniformly at random to begin with, we also have $\Pr[H = H_*] = \Pr[H = H'_*]$. The result follows from Bayes' rule. □

Lemma 4.1 reduces the problem to showing that decoding with $P_{\mathcal{C},t}$ is adaptively robust — even though $P_{\mathcal{C},t}$ contains far too many parity checks for the decoder to actually use. The following lemma is the key ingredient to showing that decoding with $P_{\mathcal{C},t}$ is adaptively robust. Recall from Section 3 that $\mathsf{bias}(z) = \frac{1}{m}\sum_{i=1}^m (-1)^{z_i}$. Note that $\mathsf{bias}(z) = 1 - 2\,\mathrm{wt}(z)/n$ and $\mathrm{wt}(z) = (1/2 - \mathsf{bias}(z)/2) \cdot n$. Recall that we use $W_t \in \mathbb{F}_2^{\binom{n}{t} \times n}$ to denote the matrix whose rows are all weight-t parity checks.

Lemma 4.2. *Let $N_{\mathcal{C},t}$ be the number of t-sparse parities consistent with $\mathcal{C}$. For any vector $x \in \mathbb{F}_2^n$ and code $\mathcal{C}$ with $N_{\mathcal{C},t} > 0$, we have that*

$$\mathsf{bias}(P_{\mathcal{C},t}x) = \frac{\binom{n}{t} \cdot 2^{-\dim(\mathcal{C})}}{N_{\mathcal{C},t}} \sum_{c \in \mathcal{C}} \mathsf{bias}(W_t(x \oplus c)).$$

Proof. First, observe that for any vector $v \in \mathbb{F}_2^n$, we have that

$$1\{v \in \mathcal{C}^\perp\} = 2^{-\dim(\mathcal{C})} \sum_{c \in \mathcal{C}} (-1)^{v \cdot c}\ .$$

Using this identity, we have the following series of equalities:

$$\begin{aligned}
N_{\mathcal{C},t} \cdot \mathsf{bias}(P_{\mathcal{C},t} x) &= \sum_{v \in \mathrm{rows}(P_{\mathcal{C},t})} (-1)^{v \cdot x} \\
&= \sum_{v \in \mathrm{rows}(W_t) \cap \mathcal{C}^\perp} (-1)^{v \cdot x} \\
&= \sum_{v \in \mathrm{rows}(W_t)} (-1)^{v \cdot x} 1\{v \in \mathcal{C}^\perp\} \\
&= \sum_{v \in \mathrm{rows}(W_t)} (-1)^{v \cdot x} \left(2^{-\dim(\mathcal{C})} \sum_{c \in \mathcal{C}} (-1)^{v \cdot c} \right) \\
&= 2^{-\dim(\mathcal{C})} \sum_{c \in \mathcal{C}} \left(\sum_{v \in \mathrm{rows}(W_t)} (-1)^{v \cdot (x \oplus c)} \right) \\
&= 2^{-\dim(\mathcal{C})} \sum_{c \in \mathcal{C}} \binom{n}{t} \mathsf{bias}(W_t(x \oplus c)) \ . \qquad \square
\end{aligned}$$

The quantity outside of the sum, $\binom{n}{t} \cdot 2^{-\dim(\mathcal{C})} / N_{\mathcal{C},t}$, can be assumed to be 1 by Fact 4.2. Therefore Lemma 4.2 allows us to reason about $\mathsf{bias}(P_{\mathcal{C},t})$, which appears to be a hopelessly unwieldy quantity, by reasoning instead about random t-sparse parities.

4.4 An adaptively robust zero-bit PRC

We will show in Theorem 6 that, if $\mathcal{C}$ is selected as the span of d random vectors, then $\mathsf{bias}(P_{\mathcal{C},t} x) \geq n^{\Omega(1)-1/2}$ for *every* $x \in \mathbb{F}_2^n$ with $\mathsf{bias}(x) = \Omega(1)$; by [CG24, Lemma 9], this also holds when $\mathcal{C}$ is selected by the key generation algorithm described above. By Lemma 4.1, it follows that any x the adversary chooses based only on knowledge of $\mathcal{C}$ will be detected with high probability over the choice of r parities from $P_{\mathcal{C},t}$.

Fact 4.2. *If $\omega(1) \leq t \leq n^{o(1)}$ and $d \leq (1 - \Omega(1)) \cdot t \log n$, then*

$$\Pr_{G \leftarrow \mathbb{F}_2^{n \times d}} \left[N_{\mathrm{Im}(G),t} = (1 \pm \mathsf{negl}(n)) \cdot \binom{n}{t} \cdot 2^{-d} \right] = 1 - \mathsf{negl}(n) .$$

Proof. The proof is a simple application of Chebyshev's inequality, and is also shown in [CG24, Lemma 11]. We present it here for completeness.

For uniformly random $G \leftarrow \mathbb{F}_2^{n \times d}$, let $X_w = 1\{wG = 0\}$ be random variables for each $w \in W_t$. Note that $N_{\mathrm{Im}(G),t} = \sum_{w \in W_t} X_w$, and for each $w \in W_t$, $\mathbb{E}[X_w] = 2^{-d}$ and $\mathsf{Var}[X_w] = 2^{-d} - 2^{-2d}$.

Letting $\alpha = \binom{n}{t} \cdot 2^{-d}$,

$$\mathbb{E}\, N_{\mathrm{Im}(G),t} = \sum_{w \in W_t} \mathbb{E}[X_w] = \alpha.$$

Furthermore, $\{X_w\}_{w \in W_t}$ are pairwise independent. So by Chebyshev's inequality,[16]

$$\begin{aligned}
\Pr\left[\left| N_{\mathrm{Im}(G),t} - \alpha \right| > \tau \right] &\leq \binom{n}{t} \cdot (2^{-d} - 2^{-2d}) / \tau^2 \\
&\leq \alpha / \tau^2.
\end{aligned}$$

[16] We use the following version of Chebyshev's inequality: for N pairwise independent random variables $X_1, \ldots, X_N$ where $\mathbb{E}\, X_1 = \cdots = \mathbb{E}\, X_N = \mu$ and $\mathsf{Var}\, X_1 = \cdots = \mathsf{Var}\, X_N = \sigma^2$, $\Pr\left[\left| \sum_{i=1}^N X_i - N\mu \right| > \tau \right] \leq N\sigma^2/\tau^2$.

Let $\tau = \alpha^{2/3}$. We have

$$\Pr\Big[N_{\mathrm{Im}(G),t} = \Big(1 \pm \alpha^{-1/3}\Big)\,\alpha\Big] \geq 1 - \alpha^{-1/3}.$$

Invoking the assumptions that $t \leq n^{o(1)}$ and $d \leq (1 - \Omega(1)) \cdot t \log n$,

$$\begin{aligned}\alpha &\geq \left(\frac{n}{t}\right)^t \cdot 2^{-d} \\ &= 2^{t \log n - d - t \log t} \\ &= 2^{t \log n - (1-\Omega(1)) \cdot t \log n - o(t \log n)} \\ &= 2^{\Omega(t \log n)},\end{aligned}$$

which is super-polynomial by the assumption that $t = \omega(1)$. This completes the proof. □

Fact 4.3. *For any vector $x \in \mathbb{F}_2^n$ and any even $t \leq \sqrt{n/2}$,*

$$\mathsf{bias}(x)^t \cdot \left(1 - \frac{2t^2/n}{\mathsf{bias}(x)^2}\right) \leq \mathsf{bias}(W_t x) \leq \mathsf{bias}(x)^t \tag{3}$$

Proof. Recall that $\mathsf{bias}(W_t x) = \mathbb{E}_{w \leftarrow W_t}(-1)^{v \cdot x}$. Let W_t^* be the distribution over $\mathbb{F}_2^n$ defined by summing t independent, random indicator vectors e_i. In other words, W_t^* is the same distribution as W_t except that we allow repeats. Let $Q_t = \Pr[\mathrm{wt}(w) = t \mid w \leftarrow W_t^*]$ be the probability that there are no repeated indices sampled under W_t^*. Note that W_t^* conditioned on having no repeats is exactly W_t; and W_t^* conditioned on having repeats is exactly W_{t-2}^*. We have

$$\begin{aligned}\mathsf{bias}(x)^t &= \mathop{\mathbb{E}}_{w \leftarrow W_t^*}(-1)^{w \cdot x} \\ &= Q_t \mathop{\mathbb{E}}_{w \leftarrow W_t}[(-1)^{w \cdot x} \mid \mathrm{wt}(w) = t] + (1 - Q_t) \mathop{\mathbb{E}}_{w \leftarrow W_t^*}[(-1)^{w \cdot x} \mid \mathrm{wt}(w) < t] \\ &= Q_t \cdot \mathsf{bias}(W_t x) + (1 - Q_t) \cdot \mathsf{bias}(x)^{t-2}.\end{aligned}$$

Rearranging and using the fact that $Q_t \geq 1 - t^2/n$, we obtain the lower bound on $\mathsf{bias}(W_t x)$ as

$$\begin{aligned}\mathsf{bias}(W_t x) &= Q_t^{-1} \cdot \mathsf{bias}(x)^t + (1 - Q_t^{-1}) \cdot \mathsf{bias}(x)^{t-2} \\ &\geq \mathsf{bias}(x)^t + \left(1 - \frac{1}{1 - t^2/n}\right) \cdot \mathsf{bias}(x)^{t-2} \\ &\geq \mathsf{bias}(x)^t - \frac{2t^2}{n} \cdot \mathsf{bias}(x)^{t-2}.\end{aligned}$$

For the last inequality, we used that t is even and $t \leq \sqrt{n/2}$ and $1 - 1/(1 - z) \geq -2z$ for $z \in [0, 1/2]$. The upper bound on $\mathsf{bias}(W_t x)$ follows by maximizing $Q_t^{-1} \cdot \mathsf{bias}(x)^t + (1 - Q_t^{-1}) \cdot \mathsf{bias}(x)^{t-2}$ over $Q_t \in [0, 1]$, using the fact that $0 \leq \mathsf{bias}(x)^t \leq \mathsf{bias}(x)^{t-2}$. □

Fact 4.4. *For any vector $x \in \mathbb{F}_2^n$ and any even $t = n^{o(1)}$,*

$$\mathsf{bias}(W_t x) \geq -n^{-(1/2 - o(1)) \cdot t}.$$

Proof. If $\mathsf{bias}(x)^2 \geq 2t^2/n$, then Fact 4.3 implies that $\mathsf{bias}(W_t x) \geq 0$.

If $\mathsf{bias}(x)^2 < 2t^2/n$, then by Fact 4.3 we still have that

$$\begin{aligned}\mathsf{bias}(W_t x) &\geq -\frac{2t^2}{n} \cdot \mathsf{bias}(x)^{t-2} \\ &\geq -\frac{2t^2}{n} \cdot \left(\frac{2t^2}{n}\right)^{t/2-1} \\ &= -\left(\frac{2t^2}{n}\right)^{t/2} \\ &\geq -n^{-(1/2-o(1))\cdot t}.\end{aligned}$$

□

Note that Theorem 6 only applies for *even* values of t.

Theorem 6. *Let $t = n^{o(1)}$ be even and $d \leq (1/2 - \Omega(1)) \cdot t \log n$. Let $G \leftarrow \mathbb{F}_2^{n \times d}$ be uniformly random and $\mathcal{C}$ be the column span of G. Then with probability $1 - \mathsf{negl}(n)$ over G,*

$$\mathsf{bias}(P_{\mathcal{C},t} e) \geq (1 - o(1)) \cdot \mathsf{bias}(e)^t \text{ for all } e \in \mathbb{F}_2^n \text{ s.t. } \mathsf{bias}(e) = \Omega(1).$$

Proof. By Lemma 4.2 and Fact 4.2, it suffices to bound

$$\sum_{c \in \mathcal{C}} \mathsf{bias}(W_t(x \oplus c)) = \mathsf{bias}(W_t x) + \sum_{c \in \mathcal{C} \setminus \{0\}} \mathsf{bias}(W_t(x \oplus c)).$$

We bound the first and second terms separately. For the first term, Fact 4.3 implies that that for any x such that $\mathsf{bias}(x) = \Omega(1)$,

$$\begin{aligned}\mathsf{bias}(W_t x) &\geq \mathsf{bias}(x)^t \cdot \left(1 - \frac{2t^2/n}{\mathsf{bias}(x)^2}\right) \\ &= (1 - O(t^2/n)) \cdot \mathsf{bias}(x)^t.\end{aligned}$$

For the second term, Fact 4.4 and the assumption that $d \leq (1/2 - \Omega(1)) \cdot t \log n$ imply that

$$\begin{aligned}\sum_{c \in \mathcal{C} \setminus \{0\}} \mathsf{bias}(W_t(x \oplus c)) &\geq -2^d \cdot n^{-(1/2-o(1))\cdot t} \\ &\geq -n^{(1/2-\Omega(1))\cdot t} \cdot n^{-(1/2-o(1))\cdot t} \\ &= -n^{-\Omega(t)}.\end{aligned}$$

□

Corollary 7. *For any $\delta \in (0, 1/2)$ and $r = n^{\Omega(1)}$, there exist $\eta = \Omega(1)$, $t = \Theta(\log n)$, and $d = \Omega(\log^2 n)$ such that the zero-bit public-key pseudorandom code $\mathsf{LDPC\text{-}PRC}_0[n, d, t, r, \eta, r^{-1/4}]$ is adaptively δ-robust (Definition 3).*

Proof. Let

- $\eta = 1/4 - \delta/2$,
- $t = \log\left(4r^{-1/4}\right)/\log(1/2 - \delta)$, and
- $d = (1/3) \cdot t \log n$.

In order to invoke Fact 4.3, t must be even, so we assume for simplicity that $4 \log\left(4r^{-1/4}\right)/\log(1/2 - \delta)$ is an even integer.

Let m', c, x, r be defined as in $\mathcal{G}^{\mathsf{robust-pk}}$ and suppose that $\mathrm{wt}(x \oplus c) \leq \delta n$. Since $\mathsf{LDPC\text{-}PRC}_0$ is a zero-bit PRC, we omit m from $\mathcal{G}^{\mathsf{robust-pk}}$. We do not omit m', however, because it could still potentially be $\perp$.

Let $c^* = c \oplus e^*$ where e^* is the ηn-sparse noise added by $\mathsf{PRC.Encode}(\mathsf{pk}; r)$. Let $e = x \oplus c^*$ be the combined error $c \oplus c^*$ and $x \oplus c$. By our choice of η and the assumption that $\mathrm{wt}(x \oplus c) \le \delta n$,

$$\begin{aligned} \mathrm{wt}(e) &\le \mathrm{wt}(x \oplus c) + \mathrm{wt}(c \oplus c^*) \\ &\le \delta n + \eta n \\ &= (1/4 + \delta/2) \cdot n \end{aligned}$$

Equivalently, this means that $\mathsf{bias}(e) \ge 1/2 - \delta = \Omega(1)$. By Theorem 6, we have that

$$\begin{aligned} \mathsf{bias}(P_{\mathcal{C},t} e) &\ge (1 - o(1)) \cdot \mathsf{bias}(e)^t \\ &\ge (1 - o(1)) \cdot (1/2 - \delta)^t \\ &= (1 - o(1)) \cdot 4r^{-1/4}, \end{aligned}$$

so $\Pr_{w \leftarrow P_{\mathcal{C},t}}[w \cdot e = 1] \le 1/2 - 2r^{-1/4}$.

Now since e is computed as a function of the public key alone, Lemma 4.1 implies that we can view H as being sampled after e. Therefore each parity check w in H has $\Pr_{w \leftarrow P_{\mathcal{C},t}}[w \cdot e = 1] \le 1/2 - 2r^{-1/4}$, and by a Chernoff bound over the parity checks in H,

$$\Pr\left[\mathrm{wt}(He) > (1/2 - r^{-1/4}) \cdot r\right] \le \mathsf{negl}(n) .$$

This means that

$$\Pr[\mathsf{PRC.Decode}(\mathsf{sk}; x) = \bot] \le \mathsf{negl}(n) ,$$

completing the proof. □

4.5 An adaptively robust single-bit PRC

Let us now state our single-bit public-key PRC construction. It is essentially a combination of two zero-bit public-key PRCs, used to encode 0 and 1 separately. The decoder checks whether *exactly one* of the zero-bit detectors accepts, in which case our decoder outputs the corresponding bit. While this construction is generic in the underlying zero-bit PRC, our proof of adaptive robustness relies on the particular structure of the LDPC-based PRCs.

Construction 2 (Single-bit public-key pseudorandom LDPC code, $\mathsf{LDPC\text{-}PRC}_1[n, d, t, r, \eta, \zeta]$)**.** *Let* $n, d, t, r : \mathbb{N} \to \mathbb{N}$ *and* $\eta, \zeta : \mathbb{N} \to [0, 1/2)$ *be efficiently-computable functions of the security parameter. We define* $\mathsf{LDPC\text{-}PRC}_0[n, d, t, r, \eta, \zeta]$ *by the following algorithms, where we leave the dependence of* n, d, t, r, η, ζ *on* λ *implicit:*

- $\mathsf{KeyGen}(1^\lambda)$*: Sample* $(H_0, G_0), (H_1, G_1) \leftarrow \mathsf{LDPC}[n, d, t, r]$ *and* $z \leftarrow \mathbb{F}_2^n$*. Output* $(\mathsf{sk} = (H_0, H_1, z), \mathsf{pk} = (G_0, G_1, z))$*.*
- $\mathsf{Encode}(1^\lambda, (G_0, G_1, z), m)$*: Sample* $u \leftarrow \mathbb{F}_2^d \setminus \{0\}$*,* $e \leftarrow \mathcal{S}_{\eta n, n}$*. Output* $G_m u \oplus z \oplus e$*.*
- $\mathsf{Decode}(1^\lambda, (H_0, H_1, z), x)$*: For* $m \in \{0, 1\}$*, if* $\mathrm{wt}(H_m(x \oplus z)) < \left(\frac{1}{2} - \zeta\right) \cdot r$ *and* $\mathrm{wt}(H_{1-m}(x \oplus z)) > \left(\frac{1}{2} - \zeta\right) \cdot r$*, output* m*; otherwise output* $\bot$*.*

In order to prove adaptive robustness of Construction 2, we need to show two things. First, that the zero-bit decoders are robust up to errors of weight $\delta n < n/4$. This follows from the same argument as in Section 4.4 (although with different parameters). Second, we need to show that the zero-bit decoders will *not* detect any codewords that are within δn Hamming distance of a codeword from the other zero-bit code. This is crucial for our decoder to know which of the zero-bit PRCs was used. We prove this second claim in Theorem 8, using our special sauce Lemma 4.2, the Johnson bound, and the fact that random linear codes are approximately unbiased.

Lemma 4.3 (Random linear codes are ε-balanced)**.** *Let $\mathcal{C} \subseteq \mathbb{F}_2^n$ be a random linear code of dimension d. Then with probability $1 - \mathsf{negl}(d)$, $\mathcal{C}$ is ε-biased for $\varepsilon = 2\sqrt{d/n}$, i.e.,*

$$|\mathsf{bias}(c)| \leq 2\sqrt{\frac{d}{n}} \text{ for all } c \in \mathcal{C} \setminus \{0\}.$$

Proof. Let G be the generator matrix for $\mathcal{C}$. For any $z \in \mathbb{F}_2^d \setminus \{0\}$,

$$\Pr_G[|\mathsf{bias}(Gz)| > \varepsilon] \leq 2e^{-\varepsilon^2 n/2}$$

by a Chernoff bound. By a union bound over z,

$$\Pr_G[\exists z \in \mathbb{F}_2^d \setminus \{0\} \text{ s.t. } |\mathsf{bias}(Gz)| > \varepsilon] \leq 2^{d-\varepsilon^2 n/2+1} = \mathsf{negl}(d)$$

if $\varepsilon = 2\sqrt{d/n}$. □

Fact 4.5. *With probability $1-\mathsf{negl}(n)$ over random linear codes $\mathcal{C}_0, \mathcal{C}_1$ of dimensions d_0, d_1 where $\omega(\log n) \leq d_0, d_1 \leq o(n)$, for every e we have*

$$\max_{\substack{c_0 \in \mathcal{C}_0 \\ c_1 \in \mathcal{C}_1 \setminus \{0\}}} |\mathsf{bias}(c_0 \oplus c_1 \oplus e)| \leq 1 - \mathsf{bias}(e) + 2\sqrt{(d_0 + d_1)/n}.$$

Proof. First we use the fact that $|\mathrm{wt}(c_0 \oplus c_1 \oplus e) - \mathrm{wt}(c_0 \oplus c_1)| \leq \mathrm{wt}(e)$ to see that

$$\begin{aligned}
\max_{\substack{c_0 \in \mathcal{C}_0 \\ c_1 \in \mathcal{C}_1 \setminus \{0\}}} |\mathsf{bias}(c_0 \oplus c_1 \oplus e)| &= \max_{c \in \mathcal{C}} |1 - 2\,\mathrm{wt}(c_0 \oplus c_1 \oplus e)/n| \\
&\leq 2\,\mathrm{wt}(e)/n + \max_{\substack{c_0 \in \mathcal{C}_0 \\ c_1 \in \mathcal{C}_1 \setminus \{0\}}} |1 - 2\,\mathrm{wt}(c_0 \oplus c_1)/n| \\
&= 1 - \mathsf{bias}(e) + \max_{\substack{c_0 \in \mathcal{C}_0 \\ c_1 \in \mathcal{C}_1 \setminus \{0\}}} |\mathsf{bias}(c_0 \oplus c_1)|.
\end{aligned}$$

Let $\mathcal{C} = \{c_0 \oplus c_1 \mid c_0 \in \mathcal{C}_0, c_1 \in \mathcal{C}_1\}$. With probability $1 - \mathsf{negl}(n)$, $\mathcal{C}_0$ and $\mathcal{C}_1$ are linearly independent, in which case $\mathcal{C}$ is a random linear code of dimension $d_0 + d_1$. So by Lemma 4.3,

$$\begin{aligned}
\max_{\substack{c_0 \in \mathcal{C}_0 \\ c_1 \in \mathcal{C}_1 \setminus \{0\}}} |\mathsf{bias}(c_0 \oplus c_1)| &\leq \max_{c \in \mathcal{C} \setminus \{0\}} |\mathsf{bias}(c)| \\
&\leq 2\sqrt{(d_0 + d_1)/n}
\end{aligned}$$

with probability $1 - \mathsf{negl}(n)$ over $\mathcal{C}_0, \mathcal{C}_1$, completing the proof. □

Lemma 4.4 (Johnson bound adapted from [GRS12, Equation 7.6])**.** *Let $\mathcal{C} \subseteq \mathbb{F}_2^n$ be a code of distance at least $(1 - \delta)n/2$. Then for any $x \in \mathbb{F}_2^n$ and any $\tau > \sqrt{\delta}$,*

$$|\{c \in \mathcal{C} : |\mathsf{bias}(x \oplus c)| \geq \tau\}| \leq \frac{1 - \delta}{\tau^2 - \delta}.$$

Theorem 8. *Suppose that $\omega(1) \leq t \leq n^{o(1)}$ and $d = \varepsilon t \log n$ for some constant $\varepsilon \in (0, 1/8)$. Let $G, G' \leftarrow \mathbb{F}_2^{n \times d}$ be uniformly random and $\mathcal{C}, \mathcal{C}'$ be the column spans of G, G'. Then with probability $1 - \mathsf{negl}(n)$ over G, G',*

$$\mathsf{bias}(P_{\mathcal{C},t}(c' \oplus e)) \leq (1 + o(1)) \cdot n^{4\varepsilon} \cdot \left(1 - \mathsf{bias}(e) + \sqrt{8d/n}\right)^t \text{ for all } c' \in \mathcal{C}' \setminus \{0\}, e \in \mathbb{F}_2^n.$$

Proof. By Lemma 4.2 (with Facts 4.2 and 4.3 applied), it suffices to bound

$$\sum_{c\in\mathcal{C}} \mathsf{bias}(c'\oplus e\oplus c)^t = \sum_{\substack{c\in\mathcal{C}\\ \left|\mathsf{bias}(c'\oplus e\oplus c)\right|\geq n^{-2\varepsilon}}} \mathsf{bias}(c'\oplus e\oplus c)^t + \sum_{\substack{c\in\mathcal{C}\\ \left|\mathsf{bias}(c'\oplus e\oplus c)\right|< n^{-2\varepsilon}}} \mathsf{bias}(c'\oplus e\oplus c)^t.$$

Since $\dim\mathcal{C}\leq d$, the term on the right is at most $2^d\cdot n^{-2\varepsilon t}$.

By Lemmas 4.3 and 4.4 and Fact 4.5, with probability $1-\mathsf{negl}(n)$ the term on the left is at most

$$\begin{aligned}\left(\frac{1-2\sqrt{d/n}}{n^{-4\varepsilon}-2\sqrt{d/n}}\right)\cdot\max_{c\in\mathcal{C}}\mathsf{bias}(c'\oplus e\oplus c)^t &\leq \left(\frac{1}{n^{-4\varepsilon}-2\sqrt{d/n}}\right)\cdot\left(1-\mathsf{bias}(e)+\sqrt{8d/n}\right)^t\\ &\leq (1+o(1))\cdot n^{4\varepsilon}\cdot\left(1-\mathsf{bias}(e)+\sqrt{8d/n}\right)^t\end{aligned}$$

for all $c'\in\mathcal{C}'\setminus\{0\}$ and $e\in\mathbb{F}_2^n$. □

Corollary 9. *For any $\delta\in(0,1/4)$ and $r=n^{\Omega(1)}$, there exist $\eta=\Omega(1)$, $t=\Theta(\log n)$, and $d=\Omega(\log^2 n)$ such that the single-bit public-key pseudorandom code $\mathsf{LDPC\text{-}PRC}_1[n,d,t,r,\eta,3r^{-1/5}/2]$ is adaptively δ-robust (Definition 3).*

Proof. Let

- $\eta=1/8-\delta/2$,
- $t=(1/5)\cdot\log r$,
- $\delta'=\log(1/(1/4+\delta))-1$,
- $\varepsilon=\min\{t\delta'/4\log n, 1/17\}$, and
- $d=\varepsilon t\log n$.

Let $\mathcal{C}_0,\mathcal{C}_1$ be the column spans of G_0,G_1, and z be the one-time pad, sampled from $\mathsf{LDPC\text{-}PRC}_1$. Let m,r,c,x,m' be defined as in $\mathcal{G}^{\mathsf{robust-pk}}$ and suppose that $\mathrm{wt}(x\oplus c)\leq\delta n$.

We need to show that x is detected under H_m and not detected under H_{1-m}. Recall from Construction 2 that this means that $\mathsf{bias}(H_m(x\oplus z))>3r^{-1/5}$ and $\mathsf{bias}(H_{1-m}(x\oplus z))<3r^{-1/5}$.

We will prove these two inequalities separately, but both cases will center on the total error e induced by the adversary (including both the ηn-sparse encoding noise, selected via r, and the δn-sparse attack). Let $c^*\in\mathcal{C}_m\setminus\{0\}$ and e be such that

$$x=c^*\oplus e\oplus z$$

and $\mathrm{wt}(e)\leq(\delta+\eta)\cdot n=(1/8+\delta/2)\cdot n$ (or equivalently, $\mathsf{bias}(e)\geq 3/4-\delta$).

Proof that $\mathsf{bias}(H_m(x\oplus z))>3r^{-1/5}$. This part is essentially the same as Section 4.4, except that we have set t to be larger because we do not want to detect beyond $1/4$ errors anymore.

By Theorem 6, we have that with probability $1-\mathsf{negl}(n)$,

$$\begin{aligned}\mathsf{bias}(P_{\mathcal{C}_m,t}(x\oplus z)) &= \mathsf{bias}(P_{\mathcal{C}_m,t}e)\\ &\geq(1-o(1))\cdot\mathsf{bias}(e)^t\\ &\geq(1-o(1))\cdot(3/4-\delta)^t\\ &=(1-o(1))\cdot 2^{t\log(3/4-\delta)}\\ &=r^{\log(3/4-\delta)/5}\\ &\geq 4r^{-1/5}\end{aligned}$$

because $\log(3/4 - \delta) > \log(1/2) = -1$. By Lemma 4.1 and a Chernoff bound, it follows that

$$\Pr\left[\mathsf{bias}(H_m(x \oplus z)) \leq 3r^{-1/5}\right] \leq \mathsf{negl}(n)\,.$$

Proof that $\mathsf{bias}(H_{1-m}(x \oplus z)) < 3r^{-1/5}$. By Theorem 8, with probability $1 - \mathsf{negl}(n)$,

$$\begin{aligned}
\mathsf{bias}(P_{\mathcal{C}_{1-m},t}(x \oplus z)) &= \mathsf{bias}(P_{\mathcal{C}_{1-m},t}(c^* \oplus e)) \\
&\leq (1 + o(1)) \cdot n^{4\varepsilon} \cdot \left(1/4 + \delta + \sqrt{8d/n}\right)^t \\
&\leq (1 + o(1)) \cdot n^{4\varepsilon} \cdot \left[(1/4 + \delta)^t + t\sqrt{8d/n}\right] \\
&= (1 + o(1)) \cdot \left[2^{4\varepsilon \log n - t\log(1/(1/4+\delta))} + n^{4\varepsilon} \cdot t\sqrt{8d/n}\right] \\
&= (1 + o(1)) \cdot \left[2^{4\varepsilon \log n - t\delta' - t} + n^{4\varepsilon} \cdot t\sqrt{8d/n}\right].
\end{aligned}$$

Recall that $\delta' = \log(1/(1/4 + \delta)) - 1$, which is positive since $\delta < 1/4$, and $\varepsilon = \min\{t\delta'/4\log n, 1/17\}$. Using $\varepsilon \leq t\delta'/4\log n$ for the left term and $\varepsilon \leq 1/17$ for the right, we have

$$\begin{aligned}
\mathsf{bias}(P_{\mathcal{C}_{1-m},t}(x \oplus z)) &\leq (1 + o(1)) \cdot \left[2^{-t} + r^{-1/4} \cdot t\sqrt{8d}\right] \\
&\leq 2r^{-1/5}.
\end{aligned}$$

By Lemma 4.1 and a Chernoff bound, it follows that

$$\Pr\left[\mathsf{bias}(H_{1-m}(x \oplus z)) \geq 3r^{-1/5}\right] \leq \mathsf{negl}(n)\,.$$

□

5 Ideal PRCs: the secret-key setting

This section is dedicated to extending the notion of *secret-key* adaptively secure PRC to what we call *Ideal PRC*. The corresponding definition is given in Section 5.1. We then show how to *generically* build such a PRC from *any* (secret-key) single-bit adaptively secure PRC — such as the one from Section 4.5 — in two-steps. First, in Section 5.2 we generically show how to boost the information rate of adaptively-secure PRC to become essentially optimal, almost without sacrificing the robustness. Then, in Section 5.3, we (also generically) show how to upgrade the secret-key adaptive security from Section 4.1 to our notion of ideal security in Section 5.1, using any PRF (which is implied by the existence of PRC, and thus does not require a new assumption). Moreover, this transformation almost preserves the information rate and the robustness, resulting in nearly optimal-rate ideal PRCs.

5.1 Definition

Consider the following security games. The goal of the adversary is to determine whether it is given oracle access to the actual detector of the PRC, or to an ideal decoding oracle.

$\mathcal{G}^{\mathsf{real}}_{\mathcal{A},\delta,\mathsf{PRC}}(1^\lambda)$:

1. The challenger samples $\mathsf{sk} \leftarrow \mathsf{PRC.KeyGen}(1^\lambda)$.
2. The adversary is allowed to make encoding and decoding queries:
   - For each encoding query m, the challenger responds with $\mathsf{PRC.Encode}(\mathsf{sk}, m)$.
   - For each decoding query x, the challenger responds with $\mathsf{PRC.Decode}(\mathsf{sk}, x)$.
3. The adversary returns a bit $\hat{b}$.

$\underline{\mathcal{G}^{\mathsf{ideal}}_{\mathcal{A},\delta}(1^\lambda)}$:

1. The challenger sets $\mathsf{transcript} = \emptyset$.
2. The adversary is allowed to make encoding and decoding queries:
   - For each encoding query m, the challenger responds with a fresh random string c and sets $\mathsf{transcript} = \mathsf{transcript} \cup \{(m, c)\}$.
   - For each decoding query x, the challenger responds with a random m such that $(m, c) \in \mathsf{transcript}$ and $\mathrm{wt}(c \oplus x) \leq \delta n$; if no such m exists, the challenger responds with $\bot$.
3. The adversary returns a bit $\hat{b}$.

Definition 5 (Ideal secret-key PRC security)**.** *We say that a secret-key pseudorandom code* PRC *satisfies* ideal δ-robust security *if, for any efficient adversary* $\mathcal{A}$*,*

$$\left|\Pr\left[\mathcal{G}^{\mathsf{real}}_{\mathcal{A},\delta,\mathsf{PRC}}(1^\lambda) = 1\right] - \Pr\left[\mathcal{G}^{\mathsf{ideal}}_{\mathcal{A},\delta}(1^\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda)\,.$$

5.2 Boosting the information rate

We recall the linear-rate PRC from [CG24]:

Construction 3 (Linear-rate public-key PRC)**.** *Let* PRC_1 *be a single-bit public-key PRC with block length* λ*. Let* $(\mathsf{Enc}, \mathsf{Dec})$ *be any error-correcting code with block length* $n \geq \lambda$ *and messages of length* k*. Let* $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^n$ *be any pseudorandom generator. We define* $\mathsf{PRC}_k[\mathsf{PRC}_1, (\mathsf{Enc}, \mathsf{Dec}), \mathsf{PRG}]$ *which is a* k*-bit public-key PRC as follows:*

- $\mathsf{KeyGen}_k(1^\lambda)$*: Sample* $\mathsf{sk}' \leftarrow \mathsf{PRC}_1.\mathsf{KeyGen}(1^\lambda)$ *and a random permutation* $\pi : [\lambda^2 + n] \to [\lambda^2 + n]$*. Output* $\mathsf{sk} = (\mathsf{sk}', \pi)$*.*
- $\mathsf{Encode}_k(\mathsf{sk}, \mathsf{m})$*: Given as input a message* $\mathsf{m} \in \{0,1\}^k$*, let* $s \leftarrow \{0,1\}^\lambda$*,* $r \leftarrow \mathsf{PRG}(s)$*, and*

  $$x \leftarrow \mathsf{PRC}_1.\mathsf{Encode}(\mathsf{pk}, r_1)||\ldots||\mathsf{PRC}_1.\mathsf{Encode}(\mathsf{pk}, r_\lambda)||\mathsf{PRG}(r) \oplus \mathsf{Enc}(\mathsf{m}).$$

  Output $\mathsf{PermBits}(x, \pi)$*.*
- $\mathsf{Decode}_k(\mathsf{sk}, c')$*: Let* $c = \mathsf{PermBits}(c', \pi^{-1})$*. Parse* $c = c_1||\ldots||c_\lambda||c_{\lambda+1}$ *as* λ *length-*λ *blocks followed by a length-*n *block. The decoder then computes* $y_i = \mathsf{PRC}_1.\mathsf{Decode}(\mathsf{sk}, y_i)$ *for all* $i \in [\lambda]$ *and lets* $y = y_1||\ldots||y_\lambda$*. It then outputs* $\mathsf{m} \leftarrow \mathsf{Dec}(\mathsf{PRG}(y) \oplus c_{\lambda+1})$*.*

Observe that asymptotically as the message length increases, the information rate of PRC_k approaches that of the underlying error-correcting code $(\mathsf{Enc}, \mathsf{Dec})$. There exist binary, linear-rate error-correcting codes that are robust to any rate of worst-case errors $\alpha \in (0, 1/4)$ — for instance, see [JST21].

Remark. *This PRC is public-key in the sense that codewords are pseudorandom even to an adversary that knows the public key. However, it is only* secret-key *adaptively robust, to an adversary that does* not *know the public key. Later, in Section 6.2, we construct a linear-rate PRC that is public-key adaptively robust.*

Theorem 10. *If* PRC_1 *and* $(\mathsf{Enc}, \mathsf{Dec})$ *are both* α *secret-key adaptively robust, then* $\mathsf{PRC}_k[\mathsf{PRC}_1, (\mathsf{Enc}, \mathsf{Dec}), \mathsf{PRG}]$ *is* $\alpha - o(1)$ *secret-key adaptively robust.*

Proof. First, observe that by pseudorandomness of PRC_1 we can replace the bits $r_1, \ldots, r_\lambda$ encoded under PRC_1 with 0^λ. Then, by pseudorandomness of PRG, we can replace $\mathsf{PRG}(r)$ with a uniformly random string.

Let $\tilde{c}$ denote the output of $\mathsf{PRC}_k.\mathsf{Encode}(\mathsf{sk}, \mathsf{m})$ with up to an $\alpha - \varepsilon$ fraction of adversarial substitutions applied, for any constant $\varepsilon > 0$. Recall that $\tilde{c}$ consists of a permutation applied to $\lambda + 1$ blocks each of length at least λ. We will show that the adversary cannot find errors that concentrate on any block; we then complete the proof by invoking the adaptive robustness of PRC_1 and $(\mathsf{Enc}, \mathsf{Dec})$.

For any c such that $(m, c) \in \mathsf{transcript}$, let $e = x \oplus c$. $\mathsf{PermBits}(e, \pi^{-1})$ consists of $\ell + 1$ blocks of length at least λ. If the permutation were to be chosen independently of e, the number of errors in block $i \in [\ell]$ is a random variable $X_i \sim \mathrm{Hyp}(\ell n' + n, (\alpha - \varepsilon)(\ell n' + n), n)$, following the hypergeometric distribution. The number of errors in the last block would be distributed as $X_{\ell+1} \sim \mathrm{Hyp}(\ell n' + n, (\alpha - \varepsilon)(\ell n' + n), n')$. By Lemma 3.1, for all $i \in [\ell + 1]$,

$$\Pr[X_i \geq \alpha n] \leq e^{-2\varepsilon^2 n} = \mathsf{negl}(\lambda).$$

By a union bound, every block would have less than an α fraction of errors with overwhelming probability.

We'll next show that if an adversary can find an error that concentrates on some block more than the above bounds, one can break pseudorandomness.

Consider an adversary $\mathcal{A}$ that is given oracle access to $\mathsf{Encode}_k(\mathsf{sk}, \cdot)$, and returns an error vector e that has at least an α fraction of errors on some block with non-negligible probability. $\mathcal{A}$ can be used to construct a distinguisher $\mathcal{B}$ breaking pseudorandomness of PRC_k as follows. $\mathcal{B}$ chooses a random permutation π. When $\mathcal{A}$ queries m to $\mathsf{Encode}_k(\mathsf{sk}, \cdot)$, $\mathcal{B}$ forwards the corresponding queries m_i to its oracle (which is either Encode_1 or the uniform distribution) to obtain responses c_i. It then lets $c = c_1 || \dots || c_{\ell+1}$ and returns $c' = \mathsf{PermBits}(c, \pi)$. When $\mathcal{B}$ obtains an error vector e from $\mathcal{A}$, it inverts the permutation and checks if e has at least an α fraction of errors on any block; if so, it knows it received PRC codewords rather than true randomness.

We've thus shown that there are less than an α fraction of errors on every block. Furthermore, these errors are chosen by an efficient adversary, since $\mathcal{A}$ and $\mathcal{B}$ together run in polynomial time. Therefore, adaptive robustness of PRC_1 implies that Decode_1 recovers every bit of r correctly. Adaptive robustness of the error-correcting code implies that m is recovered by Dec. □

5.3 The non-malleable transformation and ideal security

Let PRC be a k-bit pseudorandom code with adaptive robustness up to δ errors. Let $F : \{0,1\}^* \to \{0,1\}^\lambda \times \{0,1\}^\lambda$ be a PRF. Then there exists a secret-key PRC, say PRC^{sharp}, satisfying δ-ideal security.

$\mathsf{PRC}^{sharp}[\mathsf{PRF}, \mathsf{PRC}, \delta] = (\mathsf{KeyGen}^{sharp}, \mathsf{Encode}^{sharp}, \mathsf{Decode}^{sharp})$:

- $\mathsf{KeyGen}^{sharp}(1^\lambda)$: Sample $\mathsf{PRF.sk} \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$ and $\mathsf{PRC.sk} \leftarrow \mathsf{PRC.KeyGen}(1^\lambda)$. Output $\mathsf{sk} = (\mathsf{PRF.sk}, \mathsf{PRC.sk})$.
- $\mathsf{Encode}^{sharp}(\mathsf{sk}, \mathsf{m})$: Sample $r \leftarrow \{0,1\}^\lambda$ and let $(R_1, R_2) = F_{\mathsf{PRF.sk}}(r || \mathsf{m})$.

  Output $\mathsf{PRC.Encode}(\mathsf{PRC.sk}, r || \mathsf{m} || R_2; R_1)$.
- $\mathsf{Decode}^{sharp}(\mathsf{sk}, c)$: Compute $r || \mathsf{m} || R_2 = \mathsf{PRC.Decode}(\mathsf{PRC.sk}, c)$ and let $(R_1, \tilde{R}_2) = F_{\mathsf{PRF.sk}}(r || m)$. If

  $$\tilde{R}_2 = R_2 \text{ and } \mathrm{wt}(\mathsf{PRC.Encode}(\mathsf{PRC.sk}, r || \mathsf{m} || R_2; R_1) \oplus c) \leq \delta n,$$

  output m; otherwise, output $\perp$.

Now PRC^{sharp} behaves almost identically to PRC, except that it has a sharp decoding threshold. That is, any string that is more than δ-far from a codeword will always be rejected. This is important because it allows our simulator to simply reject all inputs that are more than δ far from any of the codewords output so far.

Theorem 11. *Let* PRC *be any k-bit pseudorandom code that is δ adaptively robust, and let* PRF *be any pseudorandom function. Then* $\mathsf{PRC}^{sharp}[\mathsf{PRF}, \mathsf{PRC}, \delta]$ *satisfies δ ideal security.*

Proof Sketch. We use a hybrid argument. Suppose that the adversary makes at most T queries to the decoding oracle. For $i \in 0, \dots, T$, define the following hybrid game. Let Encode^f and Decode^f denote the encoding and decoding algorithms of PRC^{sharp}, where its PRF is replaced with a random function f.

$\underline{\mathcal{H}^i_{\mathcal{A}, \mathsf{PRC}^{sharp}, \delta}(1^\lambda):}$

1. The challenger samples $\mathsf{sk} \leftarrow \mathsf{PRC}^{sharp}.\mathsf{KeyGen}(1^\lambda)$
2. The challenger sets $q = 1$ and $\mathsf{transcript} = \emptyset$.
3. The adversary is allowed to make encoding and decoding queries.
   - For each encoding query m:
     (a) The challenger computes $c \leftarrow \mathsf{PRC}^{sharp}.\mathsf{Encode}^f(\mathsf{sk}, m)$.
     (b) The challenger lets $\mathsf{transcript} = \mathsf{transcript} \cup \{(m, c)\}$.
     (c) The challenger responds with c.
   - For each decoding query x:
     (a) If $q \leq i$, the challenger responds with a random m such that $(m, c) \in \mathsf{transcript}$ and $\mathrm{wt}(c \oplus x) \leq \delta n$; if no such m exists, the challenger responds with $\bot$.
     (b) If $q > i$, the challenger responds with $\mathsf{PRC}^{sharp}.\mathsf{Decode}^f(\mathsf{sk}, x)$.
     (c) The challenger increments q, setting $q = q + 1$.
4. The adversary returns a bit $\hat{b}$.

$\mathcal{G}^{\mathsf{real}}$ *to* $\mathcal{H}^0$. Observe that $\mathcal{H}^0$ is identical to $\mathcal{G}^{\mathsf{real}}$, but with the PRF replaced with a random function. This is indistinguishable from $\mathcal{G}^{\mathsf{real}}$ by security of the PRF.

$\mathcal{H}^{i-1}$ to $\mathcal{H}^i$. For each $i \in [T]$, we show that the adversary's views in hybrids $\mathcal{H}^{i-1}$ and $\mathcal{H}^i$ are statistically close by showing that the response to the adversary's ith decoding query is the same in both cases.

Let x be the adversary's ith decoding query, and let $\mathsf{transcript}$ be as defined in $\mathcal{H}^i$ at the time that the adversary makes the ith decoding query. We show consider two cases, showing that in both cases the response in $\mathcal{H}^i$ is the same as in $\mathcal{H}^{i-1}$ with high probability:

1. There exists $(m, c) \in \mathsf{transcript}$ such that $\mathrm{wt}(c \oplus x) \leq \delta n$. In this case, with overwhelming probability, there is a unique such m and $\mathsf{PRC}^{sharp}.\mathsf{Decode}^f(\mathsf{sk}, x) = m$. This follows from adaptive robustness of PRC^{sharp}.
2. There is no $(m, c) \in \mathsf{transcript}$ such that $\mathrm{wt}(c \oplus x) \leq \delta n$. In this case, because of the sharp decoding threshold of PRC^{sharp}, we have $\mathsf{PRC}^{sharp}.\mathsf{Decode}^f(\mathsf{sk}, x) = \bot$ with overwhelming probability.

Proof of (1). Let x_i be the adversary's i^{th} decoding query, and consider the challenger's behavior in $\mathcal{H}^i$ in the case that $\mathrm{wt}(c \oplus x_i) \leq \delta n$ for some $(m, c) \in \mathsf{transcript}$. Let $m' = \mathsf{PRC}^{sharp}.\mathsf{Decode}^f(\mathsf{sk}, x_i)$ be the challenger's response, and observe that by adaptive robustness, $m' = m$. Note that we can invoke adaptive robustness (where the adversary is only given a single decoding query) because the responses to the decoding queries before i can be computed from the transcript alone.

In $\mathcal{H}^{i-1}$, the challenger responds with a random m_j of the (m_j, c_j) in the transcript such that $\mathrm{wt}(c_j \oplus x_i) \leq \delta n$. By the above argument, $m_j = m'$ for all such j, and therefore the challenger responds with m' in both $\mathcal{H}^i$ and $\mathcal{H}^{i-1}$.

Proof of (2). Let x_i be the adversary's i^{th} decoding query, and consider the challenger's behavior in $\mathcal{H}^i$ in the case that $\mathrm{wt}(c \oplus x_i) > \delta n$ for all $(m, c) \in \mathsf{transcript}$. The challenger first computes $r_i || m_i || R_2 = \mathsf{PRC}^{sharp}.\mathsf{Decode}^f(\mathsf{PRC}.\mathsf{sk}, x_i)$. It then computes $(R_1, \tilde{R_2}) \leftarrow f(r_i || m_i)$. Suppose for the sake of contradiction the challenger does not output $\bot$; this implies that $\tilde{R_2} = R_2$. Furthermore, letting $\tilde{x_i} = \mathsf{PRC}.\mathsf{Encode}(\mathsf{PRC}.\mathsf{sk}, r_i || m_i || R_2; R_1)$, we have that $\mathrm{wt}(\tilde{x_i} \oplus c) \leq \delta n$. If $r_i || m_i$ has not been queried to f yet, $\tilde{R_2} = R_2$ holds with only negligible probability. Therefore, $r_i || m_i$ must have been queried to f, and the challenger must have made this query in responding to one of the adversary's encoding queries. That is, the challenger must have added to the transcript $c = \mathsf{PRC.Encode}(\mathsf{PRC.sk}, r_i||m_i||R_2; R_1)$ where $(R_1, R_2) = f(r_i||m_i)$. Observe that $c = \tilde{x}_i$, which is within distance δn of x_i; we have thus arrived at a contradiction.

Therefore, in $\mathcal{H}^i$ the challenger responds with $\perp$ if x_i is far from all codewords in the transcript, which is identical to the challenger's behavior in $\mathcal{H}^{i-1}$.

$\mathcal{H}^T$ to $\mathcal{G}^{\mathsf{ideal}}$. Observe that $\mathcal{H}^T$ is identical to $\mathcal{G}^{\mathsf{ideal}}$, except that in $\mathcal{H}^T$ the challenger responds to encoding queries according to $\mathsf{PRC}^{sharp}.\mathsf{Encode}^f$. Observe that $\mathsf{PRC}^{sharp}.\mathsf{Encode}^f$ is identical to $\mathsf{PRC.Encode}$, provided that $\mathsf{PRC}^{sharp}.\mathsf{Encode}^f$ never samples the same r more than once. As some r is repeated with only negligible probability, $\mathcal{H}^T$ is indistinguishable from a hybrid $\mathcal{H}^{T'}$ where $\mathsf{PRC}^{sharp}.\mathsf{Encode}^f$ is replaced by $\mathsf{PRC.Encode}$. Now, observe that the challenger in $\mathcal{H}^{T'}$ makes only encoding (and not decoding) queries to PRC; by pseudorandomness of PRC the challenger's responses are indistinguishable from uniform randomness.

This completes the proof. □

6 CCA security: the public-key setting

This section is dedicated to extending the notion of *public-key* adaptively secure PRC to what we call *CCA-secure (public-key) PRC*. The corresponding definition is given in Section 6.1. We then show how to *generically* build such a CCA-secure PRC from *any* public-key single-bit adaptively secure PRC — such as the one from Section 4.5 — in two-steps. First, in Section 6.2 we generically show how to boost the information rate of adaptively-secure public-key PRC to become become a constant. Unfortunately, this suffers from worse — but still constant — robustness as compared to the secret-key transformation described in Section 5.2. Then, in Section 6.3, we (also generically) show how to upgrade the public-key adaptive security from Section 4.1 to our notion of CCA security in Section 6.1. This transformation preserves the information and error rates of the corresponding adaptively-secure public-key PRC. However, we are only able to prove its security in the random oracle model.

6.1 CCA security definition

We present a stronger definition of public-key PRC that captures both pseudorandomness and adaptive robustness. We refer to this security definition as CCA (Chosen Codeword Attack) security, in analogy with the related Chosen Ciphertext Attack security definition of normal encryption schemes.

Consider the following security games defined for a pseudorandom code $\mathsf{PRC} = (\mathsf{PRC.KeyGen}, \mathsf{PRC.Encode}, \mathsf{PRC.Decode})$. The goal of the adversary is to determine whether it is given oracle access to the encryption oracle or to a random codeword oracle, even in the presence of the decoding oracle. Simultaneously, robustness is ensured by modifying the decoding oracle in the "random case" to always output the message for codewords close to previously returned codewords.

$\underline{\mathcal{G}^{\mathsf{pub}}_{\mathcal{A},\delta,\mathsf{PRC}}(1^\lambda, b)\text{:}}$

1. The challenger samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PRC.KeyGen}(1^\lambda)$. It sends pk to $\mathcal{A}$.
2. The adversary is allowed to make encoding and decoding queries. Depending on the value of the bit b, the challenger responds.
   - If $b = 0$:
     - For each encoding query m, the challenger responds with $\mathsf{PRC.Encode}(\mathsf{pk}, m)$.
     - For each decoding query x, the challenger responds with $\mathsf{PRC.Decode}(\mathsf{sk}, x)$.

- If $b = 1$:
  - For each encoding query m, the challenger responds with a fresh random string c and sets $\mathsf{transcript} = \mathsf{transcript} \cup \{(m, c)\}$.
  - For each decoding query x, the challenger responds with a random m such that $(m, c) \in \mathsf{transcript}$ and $\mathrm{wt}(c \oplus x) \leq \delta n$. If no such m exists, the challenger responds with $\mathsf{PRC.Decode}(\mathsf{sk}, x)$.

3. The adversary returns a bit $\hat{b}$.

Definition 6 (CCA security for PRCs)**.** *We say that a secret-key pseudorandom code* PRC *satisfies* δ-CCA security *if, for any efficient adversary* $\mathcal{A}$*,*

$$\left| \Pr\left[\mathcal{G}^{\mathsf{pub}}_{\mathcal{A},\delta,\mathsf{PRC}}(1^\lambda, 0) = 1\right] - \Pr\left[\mathcal{G}^{\mathsf{pub}}_{\mathcal{A},\delta,\mathsf{PRC}}(1^\lambda, 1) = 1\right] \right| \leq \mathsf{negl}(\lambda) .$$

Remark. *Notice that when* $\delta = 0$*, this definition is equivalent to the (simulation-based) notion of CCA-security* with pseudorandom ciphertexts*, where the challenge ciphertext is either the encryption of the challenge message, or random. Normally, the attacker is prohibited from decrypting such challenge ciphertext, which is of course equivalent to modifying the decryption oracle to return the challenge message even in the random case.*

6.2 Boosting the information rate

We now show that one can boost an adaptively robust single-bit public-key PRC to a one with a linear information rate. This transformation incurs a significant reduction (from 1/4 to at best 1/32) in the (albeit still constant) error rate tolerated, which we do not attempt to optimize.

Let $\mathsf{PRC}_1 = (\mathsf{PRC}_1.\mathsf{KeyGen}, \mathsf{PRC}_1.\mathsf{Encode}, \mathsf{PRC}_1.\mathsf{Decode})$ be a single-bit public-key PRC with codewords of length λ that is adaptively robust up to error rate $\beta \in (0, 1/4)$. Let $\mathsf{ECC}_1 = (\mathsf{Enc}_1, \mathsf{Dec}_1)$ and $\mathsf{ECC}_2 = (\mathsf{Enc}_2, \mathsf{Dec}_2)$ be error-correcting codes from $\{0,1\}^\lambda$ to $\{0,1\}^{\ell j}$ for $\ell \geq \lambda$, and $\{0,1\}^{\lambda \ell}$ to $\{0,1\}^{\lambda \ell j}$, that are robust up to error rate $\alpha \in (0, 1/4)$. There are many standard codes which achieve this desired adaptive robustness for any message length, with j equal to a constant. We also let $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{\lambda \ell j}$ be a pseudorandom generator.

Then we define a public-key PRC, $\mathsf{PRC}_k = (\mathsf{KeyGen}_k, \mathsf{Encode}_k, \mathsf{Decode}_k)$, which for $k = \lambda \ell$ encodes k-bit messages into $2kj$-bit codewords, as follows:

$\mathsf{PRC}_k[\mathsf{ECC}_1, \mathsf{ECC}_2, \mathsf{PRC}_1, \mathsf{PRG}] = (\mathsf{KeyGen}_k, \mathsf{Encode}_k, \mathsf{Decode}_k)$:

- $\mathsf{KeyGen}_k(1^\lambda)$: Output $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PRC}_1.\mathsf{KeyGen}(1^\lambda)$.
- $\mathsf{Encode}_k(\mathsf{pk}, \mathsf{m})$: Draw a random $r \leftarrow \{0,1\}^\lambda$, and let $r' = \mathsf{Enc}_1(r)$. For each $i \in [k]$, let $x_i \leftarrow \mathsf{PRC}_1.\mathsf{Encode}(\mathsf{pk}, r'_i)$. Output
$$x_1 || \dots || x_k || \mathsf{PRG}(r) \oplus \mathsf{Enc}_2(\mathsf{m}).$$
- $\mathsf{Decode}_k(\mathsf{sk}, c)$: Parse c as k blocks of length n, $x_1 || \dots || x_k$, followed by a block y of length nk. Let $s_i = \mathsf{PRC}_1.\mathsf{Decode}(\mathsf{sk}, x_i)$ for each $i \in [k]$. Let $r = \mathsf{Dec}_1(s_1 || \dots || s_k)$. Output
$$\mathsf{Dec}_2(\mathsf{PRG}(r) \oplus y).$$

Theorem 12. *If* ECC_1 *and* ECC_2 *be error-correcting codes which are (worst-case) robust to substitutions up to rate* $\alpha \in (0, 1/4)$*, and* PRC_1 *is a pseudorandom code which is adaptively robust to substitutions up to rate* $\beta \in (0, 1/4)$*. Then* PRC_k *above is adaptively robust to substitutions, up to error rate* $\alpha\beta/2$*. Furthermore* PRC_k *has a linear information rate.*

Proof. Let $x_1, \ldots, x_k, y$ denote the blocks of a given codeword as defined in Decode_k. Recall that $x_1 || \ldots || x_k$ has length nk, and y has length nk as well.

We'll first show that we can recover r from $x_1, \ldots, x_k$. Since these blocks make up half of the codeword, the adversary can introduce a rate of $\alpha\beta$ errors here. Therefore, at most an α fraction of blocks x_i have an error rate of greater than β. By adaptive robustness of PRC_1, this $1 - \alpha$ fraction of blocks all decode correctly to r_i'. By adaptive robustness of ECC_1, decoding recovers r from the r_i''s.

We now show that given r, one can recover the message from y. Since y makes up half of the codeword, the adversary can introduce at most an $\alpha\beta < \alpha$ rate of errors to y. Therefore, adaptive robustness of ECC_2 ensures that $\mathsf{Dec}_2(\mathsf{PRG}(r) \oplus y)$ yields m.

Finally, observe that PRC_k encodes k-bit messages using codewords of length $2kj$. □

6.3 Construction and security proof

Inspired by the Fujisaki-Okamoto (FO) transformation [FO99] from semantic security to CCA security, we present a transformation from public-key adaptively robust PRC to public-key CCA-secure PRC in the random oracle model.

Construction. Let PRC be a *public-key* pseudorandom code for multi-bit messages with adaptive robustness up to δ errors. Let $\mathcal{H}_{n,m} = \{H : \{0,1\}^n \to \{0,1\}^m\}$ be a hash function family. We define a public-key pseudorandom code $\mathsf{PRC}^{\mathsf{CCA}}$, satisfying δ-CCA security, below.

- $\mathsf{KeyGen}^{\mathsf{CCA}}(1^\lambda)$: Sample $H \leftarrow \mathcal{H}_{n,m}$, where polynomials $n(1^\lambda) \geq \lambda$, $m(1^\lambda) \geq 2\lambda$ and $(\mathsf{PRC.pk}, \mathsf{PRC.sk}) \leftarrow \mathsf{PRC.KeyGen}(1^\lambda)$. Output $\mathsf{sk} = \mathsf{PRC.sk}$ and $\mathsf{pk} = (\mathsf{PRC.pk}, H)$.
- $\mathsf{Encode}^{\mathsf{CCA}}(\mathsf{pk}, \mathsf{m})$: Sample $r \leftarrow \{0,1\}^\lambda$ and let $(R_1, R_2) = H(r||\mathsf{m})$, where $R_1 \geq \lambda$ and $R_2 \geq \lambda$.

  Output $\mathsf{PRC.Encode}(\mathsf{PRC.pk}, r||\mathsf{m}||R_2; R_1)$.
- $\mathsf{Decode}^{\mathsf{CCA}}(\mathsf{sk}, c)$: Compute $r||\mathsf{m}||R_2 = \mathsf{PRC.Decode}(\mathsf{PRC.sk}, c)$ and let $(R_1, \tilde{R}_2) = H(r||m)$. If

  $$\tilde{R}_2 = R_2 \text{ and } \mathrm{wt}(\mathsf{PRC.Encode}(\mathsf{PRC.pk}, r||\mathsf{m}||R_2; R_1) \oplus c) \leq \delta n,$$

  output m; otherwise, output $\perp$.

We show that the above construction satisfies CCA security.

Theorem 13. $\mathsf{PRC}^{\mathsf{CCA}}$ *satisfies δ-CCA security.*

Proof. Suppose $\mathcal{A}$ is a probabilistic polynomial time adversary participating in the experiment $\mathcal{G}^{\mathsf{pub}}_{\mathcal{A},\delta,\mathsf{PRC}^{\mathsf{CCA}}}(1^\lambda, b)$. We establish some notation first.

- Let ℓ be the number of encoding and decoding queries made by $\mathcal{A}$; we can assume that the number of encoding and decoding queries is the same without loss of generality. Recall that $\mathcal{A}$ is allowed to intersperse the encoding and decoding queries arbitrarily.
- Denote $(m_1, \ldots, m_\ell)$ to be the sequence of (adaptive) encoding queries made by $\mathcal{A}$.
- For the i^{th} encoding query m_i, the challenger does the following: it picks $r_i \leftarrow \{0,1\}^\lambda$. It then queries (m_i, r_i) to the random oracle (RO) to obtain $H(m_i||r_i) = ((R_1)_i||(R_2)_i)$. Then it sets $c_i = \mathsf{Encode}^{\mathsf{CCA}}(\mathsf{pk}, (r_i||m_i||(R_2)_i); (R_1)_i)$. It responds to $\mathcal{A}$ with c_i. Thus, $(c_1, \ldots, c_\ell)$ is the list of responses to the encoding queries.
- Denote $Q^{\mathsf{RO}}_{Ch} = \{(m_1, r_1), \ldots, (m_\ell, r_\ell)\}$ to be the set of RO queries made by the challenger during the encoding queries.
- Denote $(c_1', \ldots, c_\ell')$ to be the sequence of (adaptive) decoding queries.

- For the i^{th} decoding query c'_i, the challenger does the following: it returns the output of $\mathsf{Decode}^{\mathsf{CCA}}(\mathsf{sk}, c'_i)$ to $\mathcal{A}$. In more detail, $\mathsf{Decode}^{\mathsf{CCA}}$ first computes $(m'_i, r'_i, (R_2)'_i) \leftarrow \mathsf{PRC.Decode}(\mathsf{sk}, c'_i)$ and then returns m'_i to $\mathcal{A}$ if and only if $(R_2)'_i = (R_2)''_i$, where $H(m_i||r_i) = ((R_1)''_i||(R_2)''_i)$. Otherwise, it returns $\perp$. Denote Q_{Ch}^{Resp} to be the set of responses to the adversarial decoding queries. That is, $Q_{Ch}^{\mathsf{Resp}} = \{m'_1, \ldots, m'_q\}$. Let Q_{Ch}^{Dec} be the set of decoded outputs of the adversarial decoding queries. That is, $Q_{Ch}^{\mathsf{Dec}} = \{(m'_1, r'_1, (R_2)'_1), \ldots, (m'_q, r'_q, (R_2)'_q)\}$. Note that Q_{Ch}^{Resp} can be obtained by considering the first components of all the elements in Q_{Ch}^{Dec}.
- Denote $Q_{\mathcal{A}}^{\mathsf{RO}} = \left\{(m''_1, r''_1), \ldots, (m''_Q, r''_Q)\right\}$ to be the set of RO queries made by the adversary.

We describe the hybrids below.

Hybrid Hyb_1: $\mathcal{G}^{\mathsf{pub}}_{\mathcal{A},\delta,\mathsf{PRC}^{\mathsf{CCA}}}(1^\lambda, 0)$.

Before describing Hyb_2, we define an event below.

> $\mathsf{NeverQueried}$: There exists $(m'_i, r'_i, (R_2)'_i)$ such that (m'_i, r'_i) has never been queried by $\mathcal{A}$ to the RO until the point where the adversary makes the i^{th} decoding query c'_i. Recall that $(m'_i, r'_i, (R_2)'_i) \in Q_{Ch}^{\mathsf{Dec}}$.

Hybrid Hyb_2: This hybrid is identical to the previous hybrid except that if $\mathsf{NeverQueried}$ happens at any point, immediately abort.

Lemma 6.1. $\Pr[\mathsf{NeverQueried}] = \mathsf{negl}(\lambda)$.

Proof. We first show that $\Pr[\mathsf{NeverQueried}]$ is negligible. Note that the probability that we can predict the output of H on x where x has never been queried before is at most $\frac{1}{2^\lambda}$. Since $\mathcal{A}$ can submit ℓ decoding queries, the probability that $\mathcal{A}$ can successfully predict the output of H is at most $\frac{\ell}{2^\lambda}$. □

Lemma 6.2. *Hybrids* Hyb_1 *and* Hyb_2 *can be distinguished by* $\mathcal{A}$ *with advantage at most* $\mathsf{negl}(\lambda)$.

Proof. Conditioned on $\mathsf{NeverQueried}$ never happening, hybrids Hyb_1 and Hyb_2 are identical. This observation along with Lemma 6.1 completes the proof. □

Hybrid Hyb_3: We first define an alternate decoding procedure, denoted by $\mathsf{AltDecode}$.

> $\mathsf{AltDecode}$: on input a codeword c and a set $\{(m''_1, r''_1), \ldots, (m''_q, r''_q), m_{i_1}, c_{i_1}, \ldots, m_{i_t}, c_{i_t}\}$,
>
> - It first computes the codewords $\{c''_1, \ldots, c''_q\}$, where $c''_i = \mathsf{PRC.Encode}(\mathsf{PRC.pk}, r''_i||m''_i||(R_2)''_i; (R_1)''_i)$ and $H(r''_i||m''_i) = ((R_1)''_i, (R_2)''_i)$.
> - It then checks[17] if $\mathrm{wt}(c \oplus c''_j) \leq \delta n$ for some $j \in [q]$. It also checks if $\mathrm{wt}(c \oplus c_{i_k}) \leq \delta n$, for some $k \in [t]$. If such a j (or k) exists then it outputs m''_j (or m_{i_k}). Otherwise, it outputs $\perp$.

This hybrid is identical to the previous hybrid except that the above alternate decoding procedure $\mathsf{AltDecode}$ is used to answer decoding queries. In more detail, whenever $\mathcal{A}$ submits the i^{th} decoding query c'_i, run $\mathsf{Altdecode}(c'_i, \{(m''_1, r''_1), \ldots, (m''_q, r''_q), m_{i_1}, c_{i_1}, \ldots, m_{i_t}, c_{i_t}\})$, where $\{(m''_1, r''_1), \ldots, (m''_q, r''_q)\}$ is the set of recorded RO queries made so far by $\mathcal{A}$ and $c_{i_1}, \ldots, c_{i_t}$ is the set of codewords generated for the messages $m_{i_1}, \ldots, m_{i_t}$ during the encoding queries made so far by $\mathcal{A}$. Denote the result to be $m^{(i)}_{\mathsf{alt}}$. Also, run the real decoding procedure $\mathsf{Decode}(\mathsf{sk}, c)$ to obtain $m^{(i)}_{\mathsf{real}}$. If $m^{(i)}_{\mathsf{alt}} \neq m^{(i)}_{\mathsf{real}}$, abort the experiment. Otherwise, return $m^{(i)}_{\mathsf{alt}}$ to $\mathcal{A}$.

Lemma 6.3. *Assuming the adaptive robustness property of* PRC, *hybrids* Hyb_2 *and* Hyb_3 *can be distinguished by* $\mathcal{A}$ *with advantage at most* $\mathsf{negl}(\lambda)$.

[17] It could be the case that c is close to more than one codeword in the set $\{c''_1, \ldots, c''_q\} \cup \{c_{i_1}, \ldots, c_{i_t}\}$. In this case, $\mathsf{AltDecode}$ picks one of the matched codewords at random.

Proof. Conditioned on $m_{\mathsf{alt}}^{(i)} = m_{\mathsf{real}}^{(i)}$, for every $i \in [\ell]$ – i.e. the outputs of the alternate and the real decoding procedures are the same – the hybrids Hyb_2 and Hyb_3 are identical. Consider the following event.

$\underline{\mathsf{UnEqual}}$: there exists $i^* \in [\ell]$ such that $m_{\mathsf{alt}}^{(i^*)} \neq m_{\mathsf{real}}^{(i^*)}$.

We claim that $\Pr[\mathsf{UnEqual}]$ is $\mathsf{negl}(\lambda)$. Suppose $\Pr[\mathsf{UnEqual}]$ is non-negligible, we violate the adaptive robustness property of PRC.

We design the following reduction that violates the adaptive robustness property of PRC:

- It samples $\widehat{i} \xleftarrow{\$} [\ell]$.
- It employs lazy sampling to simulate the RO queries made by $\mathcal{A}$.
- It duly forwards the public key it receives from the external challenger to $\mathcal{A}$.
- Encoding queries: for the i^{th} query m_i from $\mathcal{A}$, the reduction samples r_i uniformly at random and submits (m_i, r_i) to the external challenger. It receives a codeword c_i from the external challenger which it duly forwards to $\mathcal{A}$.
- Decoding queries: for the i^{th} decoding query c'_i, it first checks if $i < \widehat{i}$. If so, it employs the alternate decoding procedure $\mathsf{AltDecode}$ to respond to decoding queries. For the $(\widehat{i})^{th}$ decoding query, it checks if $m_{\mathsf{alt}}^{(\widehat{i})} \neq m_{\mathsf{real}}^{(\widehat{i})}$, where $m_{\mathsf{alt}}^{(\widehat{i})}$ and $m_{\mathsf{real}}^{(\widehat{i})}$ are as defined in the description of Hyb_3. If the check did not pass, it aborts the experiment. Otherwise, it sets $((m, r, R_2), R_1)$ as follows: (a) $c \leftarrow \mathsf{PRC}.\mathsf{Encode}(\mathsf{PRC}.\mathsf{pk}, (m, r, R_2); R_1)$, (b) $\mathrm{wt}(c, c'_{\widehat{i}}) \leq \delta n$ and, (c) $m = m_{\mathsf{alt}}^{(\widehat{i})}$. It sends $((m, r),\ (R_2),\ c'_{\widehat{i}})$ to the external challenger and ends the adaptive robustness experiment. In other words, after the $(\widehat{i})^{th}$ decoding query, the execution of the reduction ends.
- RO queries: The reduction answers consistently according to the table generated during the lazy sampling procedure.

With probability $\frac{1}{\ell}\Pr[\mathsf{UnEqual}]$, which is non-negligible by our assumption, the following events holds: $(\widehat{i})^{th}$ decoding query is the first decoding query where the event $\mathsf{UnEqual}$ holds.

This shows that the reduction violates the adaptive robustness property of PRC with non-negligible probability, which is a contradiction. □

$\underline{\textbf{Hybrid } \mathsf{Hyb}_4}$: Sample $(c_1, \ldots, c_\ell)$ instead from the uniform distribution.

Lemma 6.4. *Assuming the pseudorandomness of* PRC*, the hybrids* Hyb_3 *and* Hyb_4 *can be distinguished by* $\mathcal{A}$ *with advantage at most* $\mathsf{negl}(\lambda)$*.*

Proof. We define the following event.

$\underline{\mathsf{BadDecodingQuery}}$: If there exists i such that $(m_i, r_i) \in Q_{\mathcal{A}}^{\mathsf{RO}}$.

We first claim that the probability that $\mathsf{BadDecodingQuery}$ happens in Hyb_3 is $p = \mathsf{negl}(\lambda)$.

Suppose not; that is, let p be non-negligible. We show that we can violate the pseudorandomness of PRC. The reduction, that violates the pseudorandomness of PRC, does the following:

- It employs lazy sampling to simulate the RO queries made by $\mathcal{A}$.
- It duly forwards the public key it receives from the external challenger to $\mathcal{A}$.
- Encoding queries: for the i^{th} query m_i from $\mathcal{A}$, the reduction samples r_i uniformly at random and submits (m_i, r_i) to the external challenger. It receives a codeword c_i from the external challenger which it duly forwards to $\mathcal{A}$.

- Decoding queries: it employs the alternate decoding procedure $\mathsf{AltDecode}$ described in Hyb_3 to respond to decoding queries.
- RO queries: if the adversary ever queries (m_i, r_i), the reduction stops the execution and outputs 1. Otherwise, it answers consistently according to the table generated during the lazy sampling procedure.

If $\mathcal{A}$ finishes its execution successfully then the reduction outputs 0.

Note that the probability that the reduction outputs 1 in the case when the reduction receives pseudorandom codewords from the external challenger (as generated in Hyb_3) is non-negligible. This is due to our hypothesis that the probability that $\mathsf{BadDecodingQuery}$ happens in Hyb_3 is p, which is assumed to be non-negligible. On the other hand, the probability that the reduction outputs 1 in the case when the reduction receives uniformly random codewords from the external challenger is $\mathsf{negl}(\lambda)$. This is due to the fact that $\{r_1, \ldots, r_\ell\}$ is information-theoretically hidden from $\mathcal{A}$. Hence, the probability that the adversary queries on (m_i, r_i), for any i, is at most $\frac{\ell}{2^\lambda}$. Thus, the reduction successfully violates the pseudorandomness of PRC, a contradiction. From this we can conclude that p has to be $\mathsf{negl}(\lambda)$.

We are now ready to prove the lemma. Suppose the hybrids Hyb_3 and Hyb_4 can be distinguished by $\mathcal{A}$ with probability ε. We prove by contradiction that ε is non-negligible. Consider the following reduction:

- It employs lazy sampling to simulate the RO queries made by $\mathcal{A}$.
- It duly forwards the public key it receives from the external challenger to $\mathcal{A}$.
- Encoding queries: for the i^{th} query m_i from $\mathcal{A}$, the reduction samples r_i uniformly at random and submits (m_i, r_i) to the external challenger. It receives a codeword c_i from the external challenger which it duly forwards to $\mathcal{A}$.
- Decoding queries: it employs the alternate decoding procedure described in Hyb_3 to respond to decoding queries.
- RO queries: if the adversary ever queries (m_i, r_i), the reduction aborts. Otherwise, it answers consistently according to the table generated during the lazy sampling procedure.

After the execution of $\mathcal{A}$, the output of the reduction is set to be the output of $\mathcal{A}$.

Since the probability that $\mathsf{BadDecodingQuery}$ happens is negligible in both the hybrids Hyb_3 and Hyb_4, the probability that the reduction aborts (in both the cases) is negligible. Conditioned on the reduction never aborting, the simulation of $\mathcal{A}$ by the reduction in the case when $\{c_1, \ldots, c_\ell\}$ are pseudorandom (resp., uniform) is identical to Hyb_3 (resp., Hyb_4). Thus, the probability that the reduction violates the pseudorandomness of PRC is negligibly close to ε, which is non-negligible. This contradicts the pseudorandomness of PRC. Thus, ε is negligible, which completes the proof.

□

Hybrid Hyb_5: This is identical to $\mathcal{G}^{\mathsf{pub}}_{\mathcal{A},\delta,\mathsf{PRC}^{\mathsf{CCA}}}(1^\lambda, 1)$ except that if $\mathsf{NeverQueried}$ ever happens, abort.

Lemma 6.5. *Hybrids Hyb_4 and Hyb_5 can be distinguished by $\mathcal{A}$ with advantage at most $\mathsf{negl}(\lambda)$.*

Proof. This proof is similar to the proof of Lemma 6.3. □

Hybrid Hyb_6: $\mathcal{G}^{\mathsf{pub}}_{\mathcal{A},\delta,\mathsf{PRC}^{\mathsf{CCA}}}(1^\lambda, 1)$.

Lemma 6.6. *Hybrids Hyb_5 and Hyb_6 can be distinguished by $\mathcal{A}$ with advantage at most $\mathsf{negl}(\lambda)$.*

Proof. This proof is similar to the proof of Lemma 6.2. □

This completes the proof of Theorem 13. □

Acknowledgments. Omar Alrabiah was supported in part by a Saudi Arabian Cultural Mission (SACM) Scholarship, NSF CCF-2210823 and V. Guruswami's Simons Investigator Award. Prabhanjan Ananth was supported by NSF CNS-2329938 and NSF Career-2341004. Miranda Christ was supported by a Google CyberNYC grant, an Amazon Research Award, and NSF grants CCF-2312242, CCF-2107187, and CCF-2212233. Yevgeniy Dodis was partially supported by NSF grant CNS-2055578, and a gift from Google. Sam Gunn was partially supported by a Google PhD fellowship. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Government, Amazon, Google, or any other supporting organization.

We thank Shivam Nadimpalli for helpful conversations in the early stages of this work.

References

[Aar22] Scott Aaronson. My AI Safety Lecture for UT Effective Altruism. https://scottaaronson.blog/?p=6823, November 2022. Accessed May 2023.

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

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

[CHS24] Aloni Cohen, Alexander Hoover, and Gabe Schoenbach. Watermarking language models for many adaptive users. *IACR Cryptol. ePrint Arch.*, page 759, 2024.

[FGJ$^+$23] Jaiden Fairoze, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, and Mingyuan Wang. Publicly detectable watermarking for language models. *Cryptology ePrint Archive*, 2023.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Annual international cryptology conference*, pages 537–554. Springer, 1999.

[GG24] Surendra Ghentiyala and Venkatesan Guruswami. New constructions of pseudorandom codes. *IACR Cryptol. ePrint Arch.*, page 1425, 2024.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GM24] Noah Golowich and Ankur Moitra. Edit distance robust watermarks for language models. *IACR Cryptol. ePrint Arch.*, page 898, 2024.

[GRS12] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. Essential coding theory. *Draft available at https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book/web-coding-book.pdf*, 2(1), 2012.

[GZS24] Sam Gunn, Xuandong Zhao, and Dawn Song. An undetectable watermark for generative image models. Cryptology ePrint Archive, Paper 2024/1597, 2024.

[Hoe94] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *The collected works of Wassily Hoeffding*, pages 409–426, 1994.

[JKPT12] Abhishek Jain, Stephan Krenn, Krzysztof Pietrzak, and Aris Tentes. Commitments and efficient zero-knowledge proofs from learning parity with noise. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 663–680. Springer, 2012.

[JST21] Fernando Granha Jeronimo, Shashank Srivastava, and Madhur Tulsiani. Near-linear time decoding of ta-shma's codes via splittable regularity. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1527–1536. ACM, 2021.

[JSV24] Nikola Jovanovic, Robin Staab, and Martin T. Vechev. Watermark stealing in large language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[KGW$^+$23] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. *arXiv preprint arXiv:2301.10226*, 2023.

[KJGR21] Gabriel Kaptchuk, Tushar M. Jois, Matthew Green, and Aviel D. Rubin. Meteor: Cryptographically secure steganography for realistic distributions. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 1529–1548. ACM, 2021.

[KTHL24] Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *Trans. Mach. Learn. Res.*, 2024, 2024.

[PHZS24] Qi Pang, Shengyuan Hu, Wenting Zheng, and Virginia Smith. Attacking LLM watermarks by exploiting their strengths. *CoRR*, abs/2402.16187, 2024.

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

[ZDR19] Zachary M. Ziegler, Yuntian Deng, and Alexander M. Rush. Neural linguistic steganography. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 1210–1215. Association for Computational Linguistics, 2019.

[ZZXR24] Tong Zhou, Xuandong Zhao, Xiaolin Xu, and Shaolei Ren. Bileve: Securing text provenance in large language models against spoofing with bi-level signature. *CoRR*, abs/2406.01946, 2024.

A Related work

We are motivated by a line of work on embedding hidden patterns into the outputs of generative models. To our knowledge, the first of these works was [ZDR19] in the context of steganography using text. Later, [KJGR21] observed that the scheme of [ZDR19] could be made secure by applying pseudorandom encryption (which can be built from one-way functions). However, both of these works require full knowledge of the prompt and that there are no modifications to the text, rendering them inapplicable to watermarking.

Due to the rise of generative AI, there has been a renewed interest in watermarking. Starting with [Aar22, KGW⁺23], which focus on language models, there has emerged a line of work that embeds watermarks by modifying the randomness used in these generative algorithms. In [CGZ24], they showed that it is possible to modify the randomness of a language model such that the distribution of the watermarked model is *computationally indistinguishable* from that of the original model. This indistinguishability property, called *undetectability*, implies that the watermark does not degrade the quality of the model under any efficiently computable metric. Undetectability is identical to steganographic secrecy, and an undetectable multi-bit watermark implies stateless, robust steganography.

While the watermark of [CGZ24] achieves a very strong quality guarantee, it has only very weak *robustness* — that is, a small number of edits can remove the watermark from the text. This is because it relies on using a security-parameter-length substring of the text to seed a pseudorandom function used to embed a watermark in the remainder of the response. If any word in this substring is changed, the seed cannot be recovered at detection time and the watermark disappears. Therefore the [CGZ24] scheme is only robust against a very constrained adversary that may crop a response to a sufficiently long contiguous substring, but not replace words.

In [FGJ⁺23], they introduce a notion of publicly-detectable watermarks. While it is of course possible to publish the secret watermarking key for any scheme to make it "publicly detectable," the feature that distinguishes [FGJ⁺23] is that the watermark is *unforgeable* even by an adversary who knows the detection key. However, their notion of public detectability directly contradicts their definition of robustness — an issue that other works overcome by introducing two separate watermark detection keys [CG24, ZZXR24]. Furthermore, this scheme (even the version that uses an error-correcting code) has the same level of very-weak robustness as that of [CGZ24], and is only undetectable under the assumption that every response from the model has high entropy per fixed-length "chunk" of tokens.

Other text watermarks such as [ZALW24, KTHL24] achieve stronger robustness, to even a constant rate of edits, but at the cost of significant degradation in quality. This state of affairs suggested to many a fundamental tradeoff between quality and robustness of text watermarks.

Pseudorandom codes (PRCs) were introduced to overcome this trade-off, allowing for undetectable watermarks with robustness to even a constant rate of substitutions and random deletions [CG24]. In the same paper, they also demonstrate that PRCs immediately allow for unforgeable publicly-detectable watermarks, overcoming the aforementioned issues with [FGJ⁺23]. Since [CG24], a few works have used pseudorandom codes and their watermarking framework to construct language model watermarks [GM24, GG24], as well as an image model watermark [GZS24].

As stronger robustness of a PRC directly translates to stronger watermark robustness, some works have focused on constructing PRCs with stronger robustness guarantees. In [CHS24], they consider a notion of adaptive robustness of a watermark, where the errors are allowed to depend on previously seen watermarked responses. They prove that the scheme of [CGZ24] is adaptively robust, but as discussed above, that scheme is only very-weakly robust. They conjecture that the PRC of [CG24] is adaptively robust to even a constant rate of substitutions.

Subsequently, Golowich and Moitra [GM24] showed how to construct binary PRCs from an alternative assumption — namely, the existence of a family of $(\log n)$-juntas that is hard to learn. They then presented a powerful and generic transformation that converts any binary-alphabet PRC into a polynomial-sized-alphabet PRC with robustness to non-adaptive edits.

The most recent work on constructing PRCs is that of Ghentiyala and Guruswami [GG24]. They first show that, assuming just the existence of one-way functions, there exist PRCs with robustness to any non-adaptive channel that introduces any $o(1)$ rate of errors. They then construct PRCs under alternative notions of pseudorandomness, where the distinguisher is space-bounded or is allowed $o(1)$ distinguishing advantage. They also present an interesting variant of the codes from [CG24] that permits pseudorandomness to be based on a wider range of assumptions.

Although work on watermarks offering provable guarantees has focused largely on language models, the framework of [CG24] is applicable to generative AI more broadly. For instance, [GZS24] used PRCs to build a practical image watermark for diffusion models that is quality-preserving. Therefore, improving PRCs is an avenue for improving watermarks *in general*.