

New constructions of pseudorandom codes

Surendra Ghentiyala ^{*} Venkatesan Guruswami [†]

September 2024

Abstract

Introduced in [CG24], pseudorandom error-correcting codes (PRCs) are a new cryptographic primitive with applications in watermarking generative AI models. These are codes where a collection of polynomially many codewords is computationally indistinguishable from random, except to individuals with the decoding key. In this work, we examine the assumptions under which PRCs with robustness to a constant error rate exist.

1. We show that if both the planted hyperloop assumption introduced in [BKR23] and security of a version of Goldreich’s PRG hold, then there exist public-key PRCs for which no efficient adversary can distinguish a polynomial number of codewords from random with better than $o(1)$ advantage.
2. We revisit the construction of [CG24] and show that it can be based on a wider range of assumptions than presented in [CG24]. To do this, we introduce a weakened version of the planted XOR assumption which we call the weak planted XOR assumption and which may be of independent interest.
3. We initiate the study of PRCs which are secure against space-bounded adversaries. We show how to construct secret-key PRCs of length $O(n)$ which are *unconditionally* indistinguishable from random by $\text{poly}(n)$ time, $O(n^{1.5-\epsilon})$ space adversaries.

^{*}Cornell University. sg974@cornell.edu. This work is supported in part by the NSF under Grants Nos. CCF-2122230 and CCF-2312296, a Packard Foundation Fellowship, and a generous gift from Google. This work was done while the author was visiting the Simons Institute for the Theory of Computing.

[†]Simons Institute for the Theory of Computing and Department of EECS, University of California, Berkeley. venkatg@berkeley.edu. Research supported in part by a Simons Investigator award, and NSF grant CCF-2211972.

Contents

1	Introduction	1
1.1	PRCs and Applications	1
1.2	Watermarking large language models	2
1.3	Our results	3
1.3.1	Planted hyperloop construction	3
1.3.2	Revisiting [CG24] and planted XOR assumption	3
1.3.3	Unconditional PRCs for space-bounded adversaries	4
1.4	Further directions	4
1.5	Related work	5
2	Preliminaries	6
2.1	Notation	6
2.2	Probability and combinatorics	6
2.3	Indistinguishability and LPN	7
2.4	Pseudorandom Codes	8
3	A warmup	11
4	Planted hyperloop construction	13
4.1	The assumptions	13
4.2	The construction	15
4.3	Robustness	15
4.4	A form of soundness	17
4.5	Pseudorandomness	17
4.6	Putting it all together	17
5	The weak planted XOR construction	18
5.1	The assumption	18
5.2	Evidence $\text{XOR}_{m,t,\varepsilon}$ is a weaker assumption than $\text{XOR}_{m,t,0}$	19
5.3	The construction	21
5.4	Robustness	21
5.5	A form of soundness	22
5.6	Pseudorandomness	22
5.7	Putting it all together	22
6	PRCs for space-bounded adversaries	23

6.1	Construction	23
6.2	Robustness	23
6.3	Soundness	25
6.4	Pseudorandomness	26
6.5	Putting it all together	27
7	Perspectives	28
A	Time-space hardness of $\Theta(\log n)$-sparse LPN	31
B	Hypergeometric distribution lemma	33

1 Introduction

1.1 PRCs and Applications

An exciting recent work by Christ and Gunn [CG24] introduced the notion of pseudorandom error-correcting codes (PRCs) with the intent to watermark generative AI models. Informally, pseudorandom error-correcting codes are keyed coding schemes with the following three properties (see Definition 2.14 and Definition 2.15 for details).

1. Pseudorandomness: codewords are computationally indistinguishable from random for any algorithm which does not have the secret key.
2. Robustness: anyone with the secret key can decode corrupted codewords with overwhelming probability.
3. Soundness: any fixed $x \in \{0, 1\}^n$ has a negligible probability (over the key generation algorithm) of being decoded to a message by the decoding algorithm.

One of the beautiful insights of [CG24] is that PRCs can be used to watermark generative models if one simply reinterprets the model as a channel which corrupts the randomness it uses. Consider an abstracted polynomial time generative algorithm `Generate` that as part of its input takes in a random input seed $x \in \{0, 1\}^n$, and produces content $t \in \{0, 1\}^n$. We model an adversary trying to evade detection by a channel $\mathcal{E}' : \{0, 1\}^n \rightarrow \{0, 1\}^n$ which corrupts the content t into \tilde{t} . Furthermore, we assume that there exists an algorithm `Recover` which recovers an approximation \tilde{x} of x from an edited version of \tilde{t} . The channel $\mathcal{E} = \text{Recover} \circ \mathcal{E}' \circ \text{Generate}$ then acts as a corrupting channel for the input seed x .

Notice that if we run `Generate` seeded with a codeword from a PRC, rather than truly random x , we obtain several desirable properties.

1. Undetectability [CGZ24]: the pseudorandomness of PRC outputs guarantees that watermarked content is computationally indistinguishable from unwatermarked content. This guarantees that the quality of the outputs is not degraded by watermarking.
2. Tamper resistance: since the PRC comes with a robust decoding algorithm, we can detect PRC codewords subjected to \mathcal{E} , implying that the watermark is not removed by the tampering of content by \mathcal{E}' to the generated content t .
3. Few false positives: the soundness property guarantees that for any fixed human generated text $z_1 \dots z_n$, with overwhelming probability, the decoding algorithm will not flag it as a corrupted codeword (and thus watermarked text).

In this work, we are concerned with new constructions of pseudorandom codes. The assumptions required for the construction of pseudorandom codes in [CG24] are relatively strong (see Section 1.3.2), and were subsequently weakened in the case of secret-key PRCs to the existence of a local weak pseudorandom function family. [GM24].

We restrict ourselves to constructing zero-bit PRCs (the encoded message is always “1”, see Definition 2.17) which are robust to all channels which introduce errors at a rate of $1/2 - \varepsilon$ (for

constant ε). [CG24] show how to bootstrap such constructions into constant rate PRCs (see Definition 2.16). Furthermore, [CG24, GM24] show how to bootstrap such constructions into codes which are robust to types of errors other than just substitution errors.

1.2 Watermarking large language models

We wish to emphasize that the framework of watermarking using PRCs is not restricted to any one type of generative AI model. However, to help make the philosophy of watermarking using PRCs more concrete, we review how [CG24] instantiate a PRC based scheme for watermarking large language models (LLMs).

Imagine an abstracted model of an LLM which works over the binary alphabet and always outputs text of length n . Concretely, consider an efficiently computable function $f : \{0, 1\}^* \times \{0, 1\}^* \rightarrow [0, 1]$ which takes in the prompt and the output text so far as the input and outputs the probability $p \in [0, 1]$ that the next token will be 1. The use of the binary alphabet in f is without loss of generality since all tokens can be represented in binary. Text generation on a prompt $y \in \{0, 1\}^*$ works by iteratively sampling $z_i \leftarrow \text{Ber}(f(y, z_1 \dots z_{i-1}))$ for all $i \in [1, n]$. The final output of the LLM is then $z_1 \dots z_n$.

Let us now consider a different procedure to sample from the same distribution. We first sample $x_1 \dots x_n$, each independently from $\text{Ber}(1/2)$. To generate from the LLM on a prompt $y \in \{0, 1\}^*$, we iteratively sample z_i for $i \in [1, n]$ as follows. Let $p_i = f(y, z_1, \dots, z_{i-1})$, if $p_i \leq 1/2$, sample z_i from $\text{Ber}(2p_i x_i)$, otherwise sample z_i from $\text{Ber}(1 - (1 - x_i)(1 - p_i))$. Note that since each x_i is sampled uniformly from $\text{Ber}(1/2)$, z_i is still distributed as $\text{Ber}(p_i)$, and therefore the output distribution of the LLM on a prompt y remains unchanged from the previous example.

The key now is to create an LLM that samples $x_1 \dots x_n$ from a pseudorandom error-correcting code. We will call this new LLM the watermarking LLM. We assume that the original LLM is a polynomial time algorithm (formally, we need a family of LLMs parameterized by input length for the notion of a polynomial time algorithm to make sense, but we omit such details for the sake of exposition). Therefore, the *pseudorandomness property* guarantees that the output distribution of the watermarking LLM is computationally indistinguishable from the case when $x_1 \dots x_n$ are sampled at random, which we just saw is the same as the original LLM output distribution.

Furthermore, notice that if $0 < p_i < 1$, then $z_i = x_i$ with probability greater than $1/2$. Therefore, if many p_i are bounded away from one, the output of the watermarking LLM is relatively close to the codeword $x_1 \dots x_n$. The watermarking LLM takes the codeword $x_1 \dots x_n$ as one of its inputs and outputs $z_1 \dots z_n$, and in this way it functions as a corrupting channel. For sufficiently high entropy outputs, many p_i are sufficiently close to $1/2$, therefore $z_1 \dots z_n$ is relatively close to $x_1 \dots x_n$, and anyone with the secret key can decode $z_1 \dots z_n$, thereby confirming that the output has been watermarked. Furthermore, the LLM output $z = z_1 \dots z_n$ is also robust to corruptions by an adversary trying to evade detection since \tilde{z} will still be decoded by someone with a secret key assuming that $\Delta(z, \tilde{z})$ is small (which it will be if the adversary does not make significant changes to z). Therefore, watermarked and edited text corresponds to corrupted PRC codewords.

For a discussion of how to watermark LLM text using PRCs as well as a other application of PRCs (robust steganography), we refer the reader to [CG24].

1.3 Our results

For the purpose of watermarking, our PRCs usually need to be robust to p -bounded channels (see [Definition 2.12](#)). Informally, these are channels where an adversary can arbitrarily flip any pn bits of a codeword.

1.3.1 Planted hyperloop construction

The planted hyperloop assumption, introduced in [\[BKR23\]](#), asserts that a random 5-hypergraph is distinguishable from a random 5-hypergraph with a special $\Theta(\log n)$ size 3-hypergraph planted in it with advantage at most $o(1)$. [\[BKR23\]](#) show that if both the planted hyperloop assumption and the security of Goldreich’s PRG [\[Gol11\]](#) instantiated with the predicate $P_5(x_1, \dots, x_5) = x_1 \oplus x_2 \oplus x_3 \oplus x_4x_5$ hold, then public key cryptography exists. We show that that similar assumptions imply a type of public-key PRC.

Theorem 1.1 (informal version of [Theorem 4.11](#)). *Under the assumption used to construct public key cryptography in [\[BKR23\]](#) and $o(1)$ -pseudorandomness of Goldreich’s PRG instantiated with the $P_5(x_1, \dots, x_5) = x_1 \oplus x_2 \oplus x_3 \oplus x_4x_5$ predicate, there exist public-key PRCs robust to p -bounded channels for constant $p < 1/2$ and with $o(1)$ pseudorandomness against PPT adversaries.*

Informally, by γ pseudorandomness here, we mean that any PPT algorithm can distinguish a polynomial number of samples from random with at most γ advantage.

There are at least two ways to interpret [Theorem 4.11](#). The more obvious is simply the construction of public-key PRCs from studied cryptographic assumptions. However, one can also view it as suggesting that either the planted hyperloop assumption or the security of Goldreich’s PRG with the P_5 predicate is a surprisingly strong assumption. In particular, since the only other known construction of public-key PRCs relies on fairly strong assumptions (see [Section 1.3.2](#)), this puts the assumptions of [Theorem 4.11](#) into a select group of assumptions implying public-key PRCs. Our work offers no indication of which is the correct perspective.

1.3.2 Revisiting [\[CG24\]](#) and planted XOR assumption

In [Section 5](#), we revisit the assumptions under which [\[CG24\]](#) construct PRCs. Their construction is secure if either of the following hold

1. The planted XOR assumption and polynomial security of LPN with constant noise rate
2. $2^{O(\sqrt{n})}$ security of LPN

We revisit the first of these assumptions. While polynomial security of LPN with constant noise rate is a very well established cryptography assumption, the planted XOR assumption (introduced in [\[ASS⁺23\]](#)) is relatively new. It is therefore the most critical vulnerability in the [\[CG24\]](#) construction. Informally, the planted XOR assumption says that a random matrix $G \in \{0, 1\}^{m \times n}$ modified so that $O(\log n)$ rows xor to 0^m is computationally indistinguishable from a truly random matrix. We therefore generalize and relax the assumption to what we call the weak planted XOR assumption (see [Assumption 5.2](#)). Informally, the weak planted XOR assumption (with noise rate ε) says that

a random matrix $G \in \{0, 1\}^{m \times n}$ modified so that $O(\log n)$ rows xor to a vector v sampled from $\text{Ber}(m, \varepsilon)$ is computationally indistinguishable from a truly random matrix.

We observe that both LPN and the weak planted XOR assumption have a noise rate parameter, η and ε respectively. We show that there is a wide range of points along the ε, η parameter trade-off curve for which pseudorandom codes robust to a constant noise rate exist.

Theorem 5.9. *For efficiently computable $m = \text{poly}(n)$, $t = O(\log n)$, $\eta = o(1)$, $\varepsilon = O(\log(m)/(\eta m))$ which are functions of n and constant $p \in [0, 1/2)$, if $\text{XOR}_{m,t,\varepsilon}$ holds and $\text{LPN}[\eta]$ holds, then there exists a $(1 - \text{negl}(n), 1 - \text{negl}(n), \text{negl}(n))$ -public-key PRC which is robust to all p -bounded channels and pseudorandom against all PPT adversaries.*

One can choose to read this result as saying more about the planted XOR assumption than the construction of PRCs. Since adding noise to the planted xor assumption seems to weaken it (for which we have some minor evidence [Section 5.2](#)) and interacts nicely with the LPN assumption, the weak planted xor assumption may be the next natural variant of the planted xor assumption to study.

1.3.3 Unconditional PRCs for space-bounded adversaries

A natural and fundamental question in this area is whether we can prove the unconditional existence of PRCs (not based on cryptographic conjectures). To this end, we initiate the study of PRCs which are pseudorandom against polynomial time, space-bounded adversaries. Here we rely on the results showing that the problem of learning sparse parities (possibly with noise) is hard for space-bounded adversaries [[Raz18](#), [KRT17](#), [GKLR21](#)].

Theorem 1.2 (informal). *There exists a zero-bit PRC with codeword length $O(n)$ that is robust to error rate p for any constant $p < 1/2$ and is **unconditionally** pseudorandom against adversaries which have $O(n^{3/2-\varepsilon})$ space and $\text{poly}(n)$ time.*

Unfortunately, our scheme is unlikely to be practical for watermarking generative AI as most generative models use more than $O(n^{3/2})$ auxiliary space (where n is the size of the output of the generative model). One can therefore view this result as a first step towards practical unconditional PRCs. As the field of space-bounded cryptography progresses, one may hope we will eventually be able to construct PRCs which are pseudorandom against $O(n^5)$ space and $\text{poly}(n)$ time adversaries, which may indeed be practical for watermarking generative AI. Conversely, we believe that our scheme may already be useful for other use cases, such as robust steganography for particular types of steganographic channels (see [[CG24](#)] for details on robust steganography).

1.4 Further directions

1. The construction of public-key pseudorandom codes from unstructured assumptions is possibly the biggest question left open by this work. All known constructions rely on structured assumptions like hardness of the learning parity with noise problem or the planted hyperloop assumption. Even the construction of public-key PRCs from such a strong assumption as indistinguishably obfuscation would be new and interesting.

2. The weak planted XOR assumption introduced in [Section 5](#) merits more cryptanalytic study. Can we find more evidence that such an assumption is indeed weaker than the standard planted XOR assumption?
3. It may also be interesting to study from a theoretical perspective whether there exist a general set of generative model properties such that watermarking models with those properties using some explicit error correcting codes (from some class of constructions) rather than a PRC does not significantly degrade quality of model outputs.

1.5 Related work

The idea of pseudorandom error-correcting codes was introduced in [\[CG24\]](#) with the intent of watermarking generative AI. They constructed a binary zero-bit encryption scheme robust to the bounded adversarial substitution channel and used that to construct a binary, constant rate PRC robust to both the bounded substitution channel and the random deletion channel. Followup work by Golowich and Moitra [\[GM24\]](#) showed a construction of pseudorandom codes where the alphabet size grows polynomially in the output length of the code from a zero-bit PRC. They showed how to use such large alphabet pseudorandom codes to watermark LLM texts so that they are robust to bounded edit-distance channels (channels allowing insertions, substitutions, and deletions). Interestingly, their construction assumes the existence of a $O(\log n)$ -local weak pseudorandom function family. This is quite similar to [Section 4](#) which (among other things), assumes the security of Goldreich’s PRG, which $O(1)$ -local.

PRCs are perhaps most closely related to backdoored pseudorandom generators. Backdoored PRGs (first introduced in [\[VV83\]](#)) are pseudorandom generators where anyone with a secret key can distinguish PRG outputs from random. Zero-bit PRCs can just as well be thought of as backdoored PRGs where the mechanism to distinguish PRG outputs from random is robust to errors in its input.

The planted hyperloop construction of public-key cryptography [\[BKR23\]](#) is itself based on [\[ABW10\]](#) and [\[Gol11\]](#). These all belong to lines of work labeled expander-based cryptography which utilize or change the structure of expander graphs to build cryptographic primitives [\[OST22\]](#).

[Section 6](#) is based on [\[Raz18, KRT17, GKLR21\]](#), which show that the problem of learning sparse parities with noise is hard for space-bounded algorithms. These results are intimately connected to the area of space-bounded cryptography. In space-bounded cryptography (introduced in [\[Mau92\]](#)), it is assumed all adversaries are space-bounded (have at most, say, $o(n^2)$ space, where n is the message length). Unlike traditional cryptography, researchers have been able to prove unconditional results in the bounded storage setting [\[CM97, Din01, DQW23\]](#).

Acknowledgements

The authors would like to thank Yinuo Zhang for helpful discussion. They would also like to thank Sam Gunn and Noah Stephens-Davidowitz for reviewing early drafts of this work.

2 Preliminaries

2.1 Notation

We will use the notation $\binom{[n]}{k}$ to denote the set of all size k subsets of $[n]$. We also often use the notation $x_{[a,b]}$ to denote bits a through b (inclusive) of the string x . We write $\text{Ber}(n, \eta)$ to denote the distribution $x_1 x_2 \dots x_n$ where each bit $x_i \in \{0, 1\}$ is sampled independently from $\text{Ber}(\eta)$. We write $\text{BSC}(p)$ to denote the binary symmetric channel with crossover probability p . This is the channel where each bit is flipped with probability p and remains the same with probability $1 - p$. For $x, y \in \{0, 1\}^n$, $\Delta(x, y) = |\{i : x_i \neq y_i\}|$ is the Hamming distance between x and y . We also use $\mathcal{S}_{t,n} = \{x \in \{0, 1\}^n : |x| = t\}$ to denote the Hamming sphere of dimension n and radius t .

We will write $x_1, \dots, x_n \leftarrow \mathcal{D}$ to denote sampling x_1, \dots, x_n each independently from a distribution \mathcal{D} and also occasionally overload this notation by writing $x_1, \dots, x_n \leftarrow S$ to denote sampling x_1, \dots, x_n each independently and uniformly from the set S .

If $a \in \{0, 1\}^n$ and $b \in \{0, 1\}^m$, then $ab \in \{0, 1\}^{n+m}$ denotes the concatenation of a and b . For a matrix $G \in \{0, 1\}^{n \times m}$, $G_i \in \{0, 1\}^m$ is row i of G .

2.2 Probability and combinatorics

Definition 2.1. We say a string $a \in \{0, 1\}^n$ is δ -biased if $|\{i : a_i = 0\} - \{i : a_i = 1\}| \leq \delta n$.

Lemma 2.2. Let $p_1, \dots, p_n \in [0, 1/2]$, if $X_i \sim \text{Bern}(p_i)$, then

$$\Pr_{X_1, \dots, X_n} [X_1 \oplus \dots \oplus X_n = 0] = \frac{1}{2} \left(1 + \prod_{i=1}^n (1 - 2p_i) \right).$$

Lemma 2.3 (Chernoff Bound [Che52]). Let X_1, \dots, X_n be independent random variables, each distributed as $\text{Ber}(p)$. Let $\mu = np$, and $X = X_1 + \dots + X_n$. If $\delta \geq 0$, then

$$\Pr_{X_1, \dots, X_n} [X \geq (1 + \delta)\mu] \leq e^{-\delta^2 \mu / (2 + \delta)}.$$

If $0 < \delta < 1$, then

$$\Pr_{X_1, \dots, X_n} [X \leq (1 - \delta)\mu] \leq e^{-\delta^2 \mu / 3}.$$

We will use the same insights as [CG24] to reduce the case of p -bounded adversarial channels to the case of the hypergeometric channel. For this we need the following lemma regarding the hypergeometric distribution. Let $\text{Hyp}(N, K, n)$ denote the distribution of the number of good elements chosen when choosing n elements without replacement from a population of size N which contains K good elements.

Lemma 2.4 ([Hoe94]). Let $X \sim \text{Hyp}(N, K, n)$ and $p = K/N$. Then for any $0 < t < K/N$,

$$\Pr[X \geq (p + \varepsilon)n] \leq e^{-2\varepsilon^2 n}.$$

Lemma 2.5. *If $0 \leq t \leq m \leq n$, $X \sim \text{Hyp}(n, m, t)$, then*

$$\frac{1}{2} + \frac{1}{2} \min_{\frac{m-t}{n} \leq p_i \leq \frac{m}{n-t}} \prod_{i=1}^t (1 - 2p_i) \leq \Pr[X \text{ is even}] \leq \frac{1}{2} + \frac{1}{2} \max_{\frac{m-t}{n} \leq p_i \leq \frac{m}{n-t}} \prod_{i=1}^t (1 - 2p_i) .$$

Corollary 2.6. *If $0 \leq t \leq m \leq n$, $X \sim \text{Hyp}(n, m, t)$ and p is a value maximizing $|1 - 2p|$ subject to $(m - t)/n \leq p \leq m/(n - t)$, then*

$$\Pr[X \text{ is even}] \leq \frac{1}{2} + \frac{1}{2} |1 - 2p|^t, \quad \text{and} \quad \Pr[X \text{ is odd}] \leq \frac{1}{2} + \frac{1}{2} |1 - 2p|^t .$$

See [Appendix B](#) for proofs of [Lemma 2.5](#) and [Corollary 2.6](#).

Lemma 2.7. *Let X_1, \dots, X_Q be uniformly distributed over $[N]$.*

$$\Pr_{X_1, \dots, X_Q} [\exists i \neq j, X_i = X_j] \leq \frac{Q^2}{N} .$$

Definition 2.8. *The statistical distance (also known as the total variation distance) of two distribution X and Y on a finite domain D is defined as*

$$\Delta(X, Y) = \frac{1}{2} \sum_{z \in D} |\Pr[X = z] - \Pr[Y = z]|$$

We say two distributions X and Y are statistically indistinguishable if $\Delta(X, Y) = \text{negl}(n)$.

Fact 2.9. *Let A be a set and $B \subseteq A$. If X is uniformly distributed over A , and Y is uniformly distributed over B , then $\Delta(X, Y) = 1 - |B|/|A|$.*

2.3 Indistinguishability and LPN

For a class of functions ε , we say two distribution ensembles $\{D_n\}_{n \in \mathbb{N}}$, $\{E_n\}_{n \in \mathbb{N}}$ are ε -indistinguishable if for any probabilistic polynomial time, non-uniform adversary \mathcal{A} , there exists a function $\varepsilon' \in \varepsilon$ such that

$$\left| \Pr_{x \leftarrow D_n} [\mathcal{A}(x) = 1] - \Pr_{x \leftarrow E_n} [\mathcal{A}(x) = 1] \right| \leq \varepsilon'(n)$$

We say that $\{D_n\}_{n \in \mathbb{N}}$ and $\{E_n\}_{n \in \mathbb{N}}$ are computationally indistinguishable if they are $\text{negl}(n)$ -indistinguishable. We will often use the fact that if $\Delta(X, Y) \leq \varepsilon$, then no algorithm can distinguish between a sample from X and a sample from Y with better than ε advantage.

Assumption 2.10. *For $\eta : \mathbb{N} \rightarrow \mathbb{R}$ which is a function of n , the LPN $[\eta]$ assumption states that for all $m = \text{poly}(n)$ and all probabilistic $\text{poly}(n)$ time algorithm \mathcal{A} ,*

$$\left| \Pr_{\substack{G \leftarrow \mathbb{F}_2^{n \times m}, \\ s \leftarrow \mathbb{F}_2^m, \\ e \leftarrow \text{Ber}(n, \eta)}} [\mathcal{A}(G, Gs + e) = 1] - \Pr_{\substack{G \leftarrow \mathbb{F}_2^{n \times m}, \\ u \leftarrow \mathbb{F}_2^n}} [\mathcal{A}(G, u) = 1] \right| = \text{negl}(n)$$

For a fixed G , one LPN sample consists of sampling $s \leftarrow \mathbb{F}_2^m$ and $e \leftarrow \text{Ber}(n, \eta)$ and outputting $Gs + e$. While [Assumption 2.10](#) is stated for a single LPN sample, for a randomly sampled G , a polynomial number of LPN samples would still be computationally indistinguishable from random. This follows from a standard hybrid argument.

In our construction, we will actually rely on the following assumption, which Lemma 2 of [\[ACPS09\]](#) shows is implied by [Assumption 2.10](#).

Assumption 2.11. For $\eta : \mathbb{N} \rightarrow \mathbb{R}$ which is a function of n , the $\text{LPN}[\eta]$ assumption states that for all $m = \text{poly}(n)$ and all probabilistic $\text{poly}(n)$ time algorithm \mathcal{A} ,

$$\left| \Pr_{\substack{G \leftarrow \mathbb{F}_2^{n \times m}, \\ s \leftarrow \text{Ber}(m, \eta), \\ e \leftarrow \text{Ber}(n, \eta)}}} [\mathcal{A}(G, Gs + e) = 1] - \Pr_{\substack{G \leftarrow \mathbb{F}_2^{n \times m}, \\ u \leftarrow \mathbb{F}_2^n}} [\mathcal{A}(G, u) = 1] \right| = \text{negl}(n)$$

2.4 Pseudorandom Codes

Definition 2.12 ([\[CG24\]](#)). We say that a length-preserving binary channel $\mathcal{E} : \{0, 1\}^* \rightarrow \{0, 1\}^*$ is p -bounded if for all $n \in \mathbb{N}$, $\Pr_{x \leftarrow \{0, 1\}^n} [|\mathcal{E}(x) \oplus x| > pn] \leq \text{negl}(n)$.

Definition 2.13. The d -hypergeometric channel $\mathcal{E} : \{0, 1\}^n \rightarrow \{0, 1\}^n$ is defined as the channel \mathcal{E} which takes in x , samples $y \leftarrow \mathcal{S}_{d,n}$, and outputs $\mathcal{E}(x) = x + y$.

We now define secret and public key pseudorandom codes.

Definition 2.14 (Secret-key PRC [\[CG24\]](#)). Let Σ be a fixed alphabet. An (α, β, γ) -secret-key pseudorandom error-correcting code (abbreviated as secret-key PRC) with robustness to a channel $\mathcal{E} : \Sigma^* \rightarrow \Sigma^*$ and pseudorandomness against a class of adversaries \mathcal{C} is a triple of polynomial time randomized algorithms (KeyGen, Encode, Decode) satisfying

- (Syntax) There exists functions $\ell, n, k : \mathbb{N} \rightarrow \mathbb{N}$ such that for all $\lambda \in \mathbb{N}$, $\text{KeyGen}(1^\lambda) \in \{0, 1\}^{\ell(\lambda)}$, $\text{Encode} : \{1^\lambda\} \times \{0, 1\}^{\ell(\lambda)} \times \Sigma^{k(\lambda)} \rightarrow \Sigma^{n(\lambda)}$, and $\text{Decode} : \{1^\lambda\} \times \{0, 1\}^{\ell(\lambda)} \times \Sigma^* \rightarrow \Sigma^{k(\lambda)} \cup \{\perp\}$.

- (Error correction, or robustness) For any $\lambda \in \mathbb{N}$ and any message $m \in \Sigma^{k(\lambda)}$,

$$\Pr_{\text{sk} \leftarrow \text{KeyGen}(1^\lambda)} [\text{Decode}(1^\lambda, \text{sk}, \mathcal{E}(x)) = m : x \leftarrow \text{Encode}(1^\lambda, \text{sk}, m)] \geq \alpha$$

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

$$\Pr_{\text{sk} \leftarrow \text{KeyGen}(1^\lambda)} [\text{Decode}(1^\lambda, \text{sk}, c) = \perp] \geq \beta$$

- (Pseudorandomness) For any adversary $\mathcal{A} \in \mathcal{C}$,

$$\left| \Pr_{\text{sk} \leftarrow \text{KeyGen}(1^\lambda)} [\mathcal{A}^{\text{Encode}(1^\lambda, \text{sk}, \cdot)}(1^\lambda) = 1] - \Pr_{\mathcal{U}} [\mathcal{A}^{\mathcal{U}}(1^\lambda)] \right| = \gamma$$

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

Definition 2.15 (Public-key PRC [CG24]). Let Σ be a fixed alphabet. An (α, β, γ) -public-key pseudorandom error-correcting code (abbreviated as public-key PRC) with robustness to a channel $\mathcal{E} : \Sigma^* \rightarrow \Sigma^*$ and pseudorandomness against a class of adversaries \mathcal{C} is a triple of polynomial time randomized algorithms (KeyGen, Encode, Decode) satisfying

- (Syntax) There exists functions $\ell_{\text{Dec}}, \ell_{\text{Enc}}, n, k : \mathbb{N} \rightarrow \mathbb{N}$ such that for all $\lambda \in \mathbb{N}$, $\text{KeyGen}(1^\lambda) \in \{0, 1\}^{\ell_{\text{Dec}}(\lambda)} \times \{0, 1\}^{\ell_{\text{Enc}}(\lambda)}$, $\text{Encode} : \{1^\lambda\} \times \{0, 1\}^{\ell_{\text{Enc}}(\lambda)} \times \Sigma^{k(\lambda)} \rightarrow \Sigma^{n(\lambda)}$, and $\text{Decode} : \{1^\lambda\} \times \{0, 1\}^{\ell_{\text{Dec}}(\lambda)} \times \Sigma^* \rightarrow \Sigma^{k(\lambda)} \cup \{\perp\}$.
- (Error correction, or robustness) For any $\lambda \in \mathbb{N}$ and any message $\mathbf{m} \in \Sigma^{k(\lambda)}$,

$$\Pr_{(\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(1^\lambda)} [\text{Decode}(1^\lambda, \text{sk}, \mathcal{E}(x)) = \mathbf{m} : x \leftarrow \text{Encode}(1^\lambda, \text{pk}, \mathbf{m})] \geq \alpha$$

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

$$\Pr_{(\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(1^\lambda)} [\text{Decode}(1^\lambda, \text{sk}, c) = \perp] \geq \beta$$

- (Pseudorandomness) For any adversary $\mathcal{A} \in \mathcal{C}$,

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

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

We will say that a (α, β, γ) -public-key or (α, β, γ) -secret-key PRC scheme (KeyGen, Encode, Decode) is robust to a channel \mathcal{E} and pseudorandom/secure against \mathcal{C} if the definition [Definition 2.15](#) or [Definition 2.14](#) respectively holds given \mathcal{E} is instantiated as the channel and \mathcal{C} is instantiated as the class of adversaries. We adopt this notation of \mathcal{E} and \mathcal{C} as implicit parameters in [Definition 2.15](#) and [Definition 2.14](#) so as not to clutter the parameters but one can just as easily parameterize the definition by all relevant variables (e.g. $(\alpha, \beta, \gamma, \mathcal{E}, \mathcal{C})$ -public-key PRC).

We have expanded the definitions of secret-key PRC and public-key PRC from [CG24] by including the parameters α, β, γ in the definition and including the new implicit parameter \mathcal{C} (which is necessary to formalize security against space-bounded adversaries [Section 6](#)). If we take \mathcal{C} to be all non-uniform, probabilistic, polynomial time (PPT) algorithms, and $\alpha = 1 - \text{negl}(\lambda), \beta = 1 - \text{negl}(\lambda), \gamma = \text{negl}(\lambda)$, we recover the original definitions given in [CG24]. When \mathcal{C} and \mathcal{E} are clear from context, we will often say PRC to mean a $(1 - \text{negl}(\lambda), 1 - \text{negl}(\lambda), \text{negl}(\lambda))$ -public-key PRC or $(1 - \text{negl}(\lambda), 1 - \text{negl}(\lambda), \text{negl}(\lambda))$ -secret-key PRC.

Definition 2.16. For [Definition 2.15](#) and [Definition 2.14](#), we define $k(\lambda)/n(\lambda)$ as the rate of a PRC.

Definition 2.17. We say a PRC scheme is a zero-bit PRC scheme if the only message \mathbf{m} that is ever encrypted is 1.

The image of the decoding function of a zero-bit scheme should only be $\{1, \perp\}$ since we know that 0 is never encoded by the PRC. Informally, a zero-bit PRC requires only that we distinguish corrupted PRC outputs from strings which are not PRC outputs. We will focus on zero-bit PRCs since when \mathcal{C} is all PPT algorithms, [CG24] shows that the existence of a zero-bit secret-key or public-key PRC implies the existence of a secret-key or public-key PRC respectively which has essentially the same robustness as the original but a worse rate. See [CG24] for a formal statement.

Say we have a zero-bit encryption scheme where corrupted codewords are identified as such with probability $\alpha(\lambda)$, random words are identified as codewords with probability $\alpha(\lambda) - 1/\text{poly}(n)$, and any polynomial number of codewords are γ -indistinguishable from random. This is not quite a PRC since we do not have the soundness property. However, our next lemma shows that we can use such a scheme to construct a $(1 - \text{negl}(\lambda), 1 - \text{negl}(\lambda), \gamma)$ zero-bit PRC.

Lemma 2.18. *Suppose that there exist PPT algorithms $(\text{KeyGen}, \text{Encode}, \text{Decode})$ such that*

1. *There exists functions $\ell_{\text{Dec}}, \ell_{\text{Enc}}, n, k : \mathbb{N} \rightarrow \mathbb{N}$ such that for all $\lambda \in \mathbb{N}$, $\text{KeyGen}(1^\lambda) \in \{0, 1\}^{\ell_{\text{Dec}}(\lambda)} \times \{0, 1\}^{\ell_{\text{Enc}}(\lambda)}$, $\text{Encode} : \{1^\lambda\} \times \{0, 1\}^{\ell_{\text{Enc}}(\lambda)} \times \{1\} \rightarrow \Sigma^{n(\lambda)}$, and $\text{Decode} : \{1^\lambda\} \times \{0, 1\}^{\ell_{\text{Dec}}(\lambda)} \times \Sigma^* \rightarrow \{1, \perp\}$.*

2. $n(\lambda) = \text{poly}(\lambda)$.

3. *For every $d \leq p \cdot n(\lambda)$, d -hypergeometric channel \mathcal{E} , and a $1 - \text{negl}(\lambda)$ fraction of keys $(\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(1^\lambda)$,*

$$\Pr_{\mathcal{E}}[\text{Decode}(1^\lambda, \text{sk}, \mathcal{E}(x)) = 1 : x \leftarrow \text{Encode}(1^\lambda, \text{pk}, 1)] \geq \alpha(\lambda)$$

where the randomness is over the randomness of the encoding algorithm and the errors of \mathcal{E} .

4. *There exists a $\delta(n) = 1/\text{poly}(n)$ where $\alpha(\lambda) - \delta(\lambda) \geq 1/\text{poly}(\lambda)$ such that for a $1 - \text{negl}(\lambda)$ fraction of keys $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(1^\lambda)$,*

$$\Pr_{x \leftarrow \{0,1\}^n}[\text{Decode}(1^\lambda, \text{sk}, x) = 1] \leq \delta(\lambda) .$$

5. *For any $q = \text{poly}(\lambda)$, $X_1, \dots, X_q \leftarrow \text{Enc}(1^\lambda, \text{pk}, 1)$ is γ -indistinguishable from $Y_1, \dots, Y_q \leftarrow \{0, 1\}^{n(\lambda)}$.*

Then for every constant $\varepsilon > 0$, there exists of a zero-bit $(1 - \text{negl}(\lambda), 1 - \text{negl}(\lambda), \gamma(\lambda))$ -public-key PRC robust to any $(p - \varepsilon)$ -bounded channel and pseudorandom against any PPT adversary.

Proof. Let $\varepsilon > 0$ be an arbitrarily small constant. Say $\alpha(\lambda) - \delta(\lambda) \geq 1/\lambda^c$ for some constant c and sufficiently large n , and let $t = \lambda^{100c}/\delta(\lambda)$. We now construct a $(1 - \text{negl}(\lambda), 1 - \text{negl}(\lambda), \gamma(\lambda))$ -public-key PRC with key generation, encoding, and decoding functions KeyGen' , Encode' , Decode' respectively.

- $\text{KeyGen}'(1^\lambda)$: Sample $(\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(1^\lambda)$, $z_1, \dots, z_t \leftarrow \{0, 1\}^{n(\lambda)}$, and a random permutation $\pi : [tn] \rightarrow [tn]$. Output $(\text{sk}' = (\text{sk}, z_1, \dots, z_t, \pi), \text{pk}' = (\text{pk}, z_1, \dots, z_t, \pi))$.
- $\text{Encode}'(1^\lambda, (\text{pk}, z_1, \dots, z_t, \pi), 1)$: Let $a_i \leftarrow \text{Encode}(1^\lambda, \text{pk}, 1)$ for $i \in [1, t]$. Output $\pi((a_1 \oplus z_1) \parallel \dots \parallel (a_t \oplus z_t))$.

- $\text{Decode}'(1^\lambda, (\text{sk}, z_1, \dots, z_t, \pi), x)$: Decompose $\pi^{-1}(x) \in \{0, 1\}^{nt}$ into $\tilde{a}_1 || \dots || \tilde{a}_t$ with $\tilde{a}_i \in \{0, 1\}^{n(\lambda)}$ and let $a_i = \tilde{a}_i \oplus z_i$. Let $w_i = 1$ if and only if $\text{Decode}(1^\lambda, \text{sk}, a_i) = 1$. Output 1 if $\sum_{i=1}^t w_i \geq t \cdot \frac{\alpha(n) + \delta(n)}{2}$ and output \perp otherwise.

Observe that all the algorithms needed to specify our new PRC scheme $\text{PRC}' = (\text{KeyGen}', \text{Encode}', \text{Decode}')$ inherit $\text{poly}(\lambda)$ running times from $(\text{KeyGen}, \text{Encode}, \text{Decode})$.

Consider a codeword $x \in \{0, 1\}^{tn}$ generated by Encode' and let $x' = \mathcal{E}(x)$. Because of the shifts z_1, \dots, z_t and permutation π applied to the codeword, for the purpose of showing robustness, we can assume without loss of generality that that Encode' and Decode' do not apply the shift z or the permutation π and \mathcal{E} samples m values without replacement from some distribution with support on $[1, (p - \varepsilon)tn]$ and then distributes m errors randomly into the tn coordinates of the codeword x . One way to imagine sampling the errors \mathcal{E} introduces is by sampling from some joint distribution (P_1, \dots, P_t) where P_i represents represents the number of errors to introduce into the i th block of n bits, and then flipping P_i bits of the i th block of n bits randomly. Let us consider a fixed $m \in [1, (p - \varepsilon)tn]$, notice that the number of errors in the i th section of n bits is exactly distributed as $\text{Hyp}(nt, m, n)$. By [Lemma 2.4](#), $P_i > pn$ with $\text{negl}(n) = \text{negl}(\lambda)$ probability. By union bound, there is a $1 - \text{negl}(\lambda)$ probability that $P_i \leq pn$ for all $i \in [1, t]$. Therefore, we can assume without loss of generality that the errors in each block of n bits come from a p -hypergeometric channel.

We now show robustness of this new scheme. Notice that during the computation of $\text{Decode}'(1^\lambda, (\text{sk}, z_1, \dots, z_t), x')$, each $w_i = 1$ independently with probability $\alpha(n)$ by [Item 3](#) and since $P_i \leq pn$. Furthermore, $t \cdot ((\alpha(\lambda) + \delta(\lambda))/2) = t(\alpha(\lambda) - (\alpha(\lambda) - \delta(\lambda))/2) = t(\alpha(\lambda) - 1/(2\lambda^c))$. So, by [Lemma 2.3](#), the probability that $\sum_{i=1}^t w_i < t \cdot \frac{\alpha(\lambda) + \delta(\lambda)}{2}$ is at most $e^{-(1/(2\lambda^c))^2 \alpha(\lambda)t/3} \leq e^{-(1/(2\lambda^c))^2 \delta(n)t/3} = \text{negl}(\lambda)$ by our choice of t . Therefore, there is a $1 - \text{negl}(\lambda)$ chance that Decode' outputs 1.

We now show soundness of the new scheme. Consider a fixed word $x \in \{0, 1\}^{tn}$. If we sample $(\text{sk}, z_1, \dots, z_t)$ and run $\text{Decode}'(1^\lambda, (\text{sk}, z_1, \dots, z_t), x) = \perp$, the probability $w_i = 0$ is exactly the probability that $\text{Decode}(1^\lambda, s_i, \tilde{a}_i \oplus z) = \perp$. This probability is $\delta(\lambda)$ by assumption since $\tilde{a}_i \oplus z$ is a uniformly random value in $\{0, 1\}^n$. Therefore, by [Lemma 2.3](#), the probability that $\sum_{i=1}^t w_i \geq t \cdot \frac{\alpha(\lambda) + \delta(\lambda)}{2} = t \cdot (\delta(\lambda) + (\alpha(\lambda) - \delta(\lambda))/2) = t \cdot (\delta(\lambda) + 1/(2\lambda^c))$ is at most $e^{-(1/(2\lambda^c))^2 \delta(\lambda)t/3} = \text{negl}(\lambda)$ (where the last equality follows from our choice of t). Therefore there is a $1 - \text{negl}(\lambda)$ chance that Decode' outputs \perp .

Pseudorandomness of our new scheme follows immediately from the pseudorandomness of the old scheme since the new scheme simply consists of codewords from the old scheme concatenated together (with a shift z and permutation π applied on top). \square

Just as in [\[CG24\]](#), for the remainder of the paper, we will identify n with the security parameter. This may be confusing since when λ is the security parameter, we say $n(\lambda)$ is the length of the code. However, letting n be the security parameter brings our notation in line with the works most closely related to our own [\[BKR23, CG24, Raz18\]](#).

3 A warmup

Here we give informal an description of a zero-bit PRC schemes with is only robust to $o(n)$ errors and pseudorandom against any PPT adversary. Our primary purpose for presenting these schemes

is to build intuition around which types of assumptions are good for building PRCs. Very similar PRF based constructions have already been described in [CG24, CGZ24, KGW⁺23, Arr22]. Though we note that those seem to achieve robustness to any $o(1/\log n)$ error rate, whereas the following scheme is robust to any $o(1)$ error rate.

We will examine a simple construction of PRCs, which, for *any* $\tau(n) = \omega(1)$, is robust against $\text{BSC}(1/\tau(n))$. We choose $\text{BSC}(1/\tau(n))$ instead of $(1/\tau(n))$ -bounded channels only for ease of presentation. The existence of one-way functions implies the existence of a keyed pseudorandom function family $f_k : \{0, 1\}^{\sqrt{\tau(n)} \log(n)} \rightarrow \{0, 1\}^n$ using the GGM construction [GGM86]. Let $t = n^3$. The key generation algorithm selects the key $k \leftarrow \{0, 1\}^n$, the encoding algorithm samples $x_1, \dots, x_t \leftarrow \{0, 1\}^{\sqrt{\tau(n)} \log(n)}$ and outputs $x_1 \| f_k(x_1) \| \dots \| x_t \| f_k(x_t)$, and the decoding algorithm on an input $\tilde{x}_1 \| \tilde{y}_1 \| \dots \| \tilde{x}_t \| \tilde{y}_t$ for $\tilde{x}_i \in \{0, 1\}^{\sqrt{\tau(n)} \log(n)}$, $\tilde{y}_i \in \{0, 1\}^n$ outputs 1 if $\Delta(f_k(\tilde{x}_i), \tilde{y}_i) \leq n/10$ for any $i \in [1, t]$ and \perp otherwise.

ROBUSTNESS. Imagine subjecting a codeword $x_1 \| f_k(x_1) \| \dots \| x_t \| f_k(x_t)$ from this scheme to $\text{BSC}(p)$ for $p = 1/\tau(n)$. As long as there exists a $i \in [t]$ such that the bits in x_i remain unchanged and the bits of $f_k(x_i)$ suffer less than $n/10$ flips, the corrupted codeword will be decoded to zero. For a fixed i , the probability that the bits of x_i remain unchanged is

$$(1-p)^{\log(n)\sqrt{\tau(n)}} = \left(\left(1 - \frac{1}{\tau(n)}\right)^{\log(n)\tau(n)} \right)^{\frac{1}{\sqrt{\tau(n)}}} \geq \left(\frac{1}{n^2}\right)^{\frac{1}{\sqrt{\tau(n)}}}$$

for sufficiently large n . The probability that every x_i is changed is at most

$$\left(1 - \frac{1}{n\sqrt{\tau(n)}}\right)^t = \text{negl}(n)$$

by our choice of t . Furthermore, for any fixed $i \in [t]$, the probability that more than $n/10$ corruptions occur in bits $f_k(x_i)$ is $\text{negl}(n)$ by [Lemma 2.3](#). By the union bound, there is a $1 - \text{negl}(n)$ chance that there exists some $i \in [t]$ such that x_i is unchanged, and $f_k(x_i)$ suffers fewer than $n/10$ bit-flips.

SOUNDNESS. Suppose the scheme were not sound, and there existed a series fixed string (parameterized by n) $x_1^* \| y_1^* \| \dots \| x_t^* \| y_t^*$ where $x_i^* \in \{0, 1\}^{\sqrt{\tau(n)} \log(n)}$, $y_t^* \in \{0, 1\}^n$ which were decoded to 1 with non-negligible probability (where the probability is over k). Notice that for a truly random function f , for any $i \in [1, t]$, $\Delta(f(x_i^*), y_i^*) \leq n/10$ with $\text{negl}(n)$ probability (by [Lemma 2.3](#)). Now consider the polynomial time distinguisher $\mathcal{A}(1^n)$ which when given oracle access to a function $f' : \{0, 1\}^{\sqrt{\tau(n)} \log(n)} \rightarrow \{0, 1\}^n$ outputs 1 if and only if there exists an $i \in [t]$ such that $\Delta(x_i^*, f'(x_i^*)) \leq n/10$. Notice that by assumption, if f' is a PRF, then \mathcal{A} outputs 1 with non-negligible probability, and if f' is truly random, it outputs 1 with negligible probability. Therefore, \mathcal{A} distinguishes between f_k and a truly random function with non-negligible probability. This contradicts the fact that f_k is a PRF. We have arrived at a contradiction, so the scheme must be sound.

PSEUDORANDOMNESS. We will show that the outputs of this scheme are pseudorandom against PPT adversaries by applying the hybrid lemma. Let $q = \text{poly}(t) = \text{poly}(n)$ and consider the following distributions:

1. $(x_1 || f_k(x_1) || \dots || x_q || f_k(x_q))$ where $k, x_1, \dots, x_q \leftarrow \{0, 1\}^n$
2. $(x_1 || f(x_1) || \dots || x_q || f(x_q))$ where f is a random function and $x_1, \dots, x_q \leftarrow \{0, 1\}^n$
3. $(x_1 || f(x_1) || \dots || x_q || f(x_q))$ where f is a random function and $x_1, \dots, x_q \leftarrow \{0, 1\}^n$ conditioned on all x_i being distinct
4. $(x_1 || y_1 || \dots || x_q || y_q)$ where $x_1, \dots, x_q, y_1, \dots, y_q \leftarrow \{0, 1\}^n$ conditioned on all x_i being distinct
5. $(x_1 || y_1 || \dots || x_q || y_q)$ where $x_1, \dots, x_q, y_1, \dots, y_q \leftarrow \{0, 1\}^n$

Distributions 1 and 2 are computationally indistinguishable by the pseudorandomness of f_k . Distributions 2 and 3 are statistically indistinguishable by [Fact 2.9](#) since conditioning on x_i being distinct only eliminates $t^2/2^{\log(n)}\sqrt{t(n)} = \text{negl}(n)$ fraction of possibilities ([Lemma 2.7](#)). Distribution 3 and 4 are the same. Distribution 4 and 5 are statistically indistinguishable by the same reasoning as showed distribution 2 and 3 are indistinguishable. Therefore, by the hybrid lemma, distributions 1 and 5 are indistinguishable. Thus, the scheme satisfies the pseudorandomness property.

We see that the minimal cryptographic assumption of one-way functions lets us achieve robustness to any $o(1)$ error rate. This sets a baseline and tells us that for a scheme to be considered non-trivial, it must be robust to a constant error rate. The critical weakness of the PRF construction is that for a codeword to be decoded correctly, all $\omega(\log n)$ bits of some x_i must remain intact. One approach to fix this (used to construct secret-key PRCs in [\[GM24\]](#)) is to use a local weak PRF family so that if that $\Delta(x, x')$ being small implies that $\Delta(f(x), f'(x))$ is small. A different approach is to aim for a scheme in which the decoding algorithm looks at a small number of bits of the codeword. In particular, if the decoding algorithm only looks at $O(\log n)$ of the the codeword, we may be able to achieve robustness to a constant error rate. This is the intuition that guides all of our upcoming schemes. This is also the approach guiding the scheme proposed in [\[CG24\]](#).

4 Planted hyperloop construction

4.1 The assumptions

We begin by reviewing the assumptions used by [\[BKR23\]](#) to construct public-key cryptography. All hypergraphs are assumed to have ordered hyperedges (a hyperedge is an ordered tuple of vertices rather than a set of vertices).

Definition 4.1. *A hyperloop is a 3-hypergraph where each vertex has degree two and we define the size of a hyperloop as the number of hyperedges it contains.*

The construction of [\[BKR23\]](#) plants $t = 2^{\Theta(\ell)}$ hyperloops S_1, \dots, S_t of size $\ell = O(\log n)$ into a random hypergraph to create a 5-hypergraph H . The secret key is the set of t hyperloops and the public key is H . We now formally present this construction.

Construction 4.2 ([\[BKR23\]](#)). *Let L_0 be a fixed hyperloop of size $\ell = O(\log n)$. H is sampled as follows. Let:*

1. L be the union of $t = 2^{\Theta(\ell)}$ vertex-disjoint copies of L_0 ,

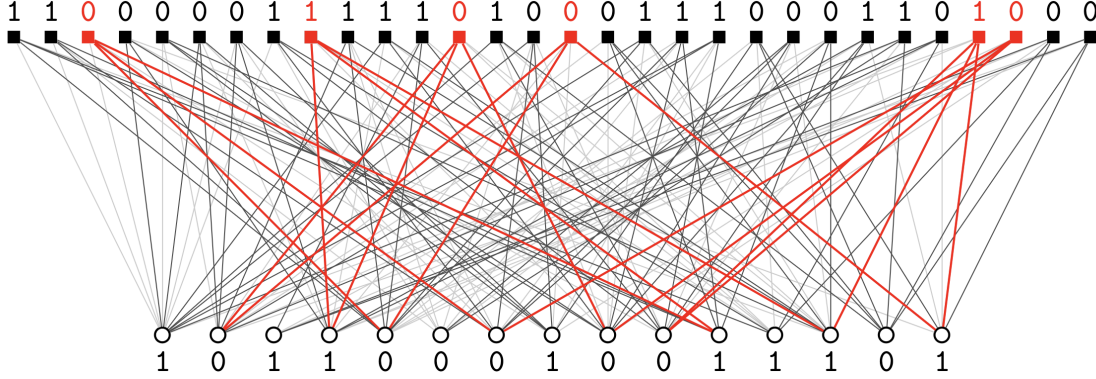


Figure 1: A public key and PRG output with a single planted hyperloop L_0 . The secret key is marked in red (from [BKR23]).

2. Q be a random 3-hypergraph with n vertices and $m = n^{3/2-\delta}$ hyperedges,
3. $P = Q \cup L$ where L is planted on a random subset of the vertices of Q ,
4. H is obtained by randomly adding 2 vertices to each hyperedge in P (where those 2 vertices will be the last two in the ordered hyperedge)

The public key is the 5-hypergraph H and the secret key is S_1, \dots, S_t where $S_i \subseteq \{1, \dots, m\}$ are they hyperedges corresponding to the i^{th} planted copy of L_0 .

We refer to any hypergraph generated using [Construction 4.2](#) as a planted hyperloop graph. This leads us to our first assumption.

Assumption 4.3. For a sufficiently small constant δ , $m = n^{1.5-\delta}$, $\ell = 0.36 \log n$, and $t = n^{0.75-\delta}$, P and Q are $o(1)$ -indistinguishable in $n^{O(1)}$ time.

The planted hyperloop assumption as stated in [BKR23] is the same as [Assumption 4.3](#) with the exception that it assumes $(1 - \Omega(1))$ -indistinguishably rather than $o(1)$ -indistinguishably. However, in the following statement, [BKR23] indicates that $o(1)$ -indistinguishably is also a fair assumption:

Our security argument applies to distinguishers of any *constant* advantage.

The fact that the security arguments of [BKR23] apply to distinguishers of *any* constant advantage rather than for some fixed constant advantage which is less than one leads us to believe that the assumption of $o(1)$ -indistinguishably in [Assumption 4.3](#) is indeed fair. If one wishes to use the more conservative assumption of $1 - \Omega(1)$ -indistinguishability for [Assumption 4.3](#), [Theorem 4.11](#) holds with $(1 - \Omega(1))$ -indistinguishability rather than $o(1)$ -indistinguishability.

We note $(1 - \Omega(1))$ -indistinguishability is a perfectly fine assumption for [BKR23] since they are able to amplify this to $\text{negl}(n)$ -indistinguishability using standard amplification techniques. However, applying such amplification techniques would mean that our PRC would not be robust to a constant error rate anymore. So we must make do with worse than $\text{negl}(n)$ -indistinguishability.

Remark 4.4. Since $\ell = O(\log n)$, a brute-force $2^{O(\log^2(n))}$ time algorithm can distinguish P from Q by searching for the implanted hyperloop. This will imply that [Construction 4.6](#) will not be

secure against $2^{O(\log^2(n))}$ time adversaries. Indeed, to our knowledge, all PRC are vulnerable to quasi-polynomial time adversaries.

Hypergraphs with certain parameters (including planted hyperloop hypergraphs) can be used as PRGs. To show how, we review Goldreich’s PRG. Fix the predicate $P_5(x_1, \dots, x_5) = x_1 \oplus x_2 \oplus x_3 \oplus x_4 x_5$. For an n vertex, m hyperedge, 5-hypergraph H , we define the PRG $F_H : \{0, 1\}^n \rightarrow \{0, 1\}^m$ as follows. On an input x , the bits of x are projected onto the vertices of H , and bit i of $F_H(x)$ is given by applying P_5 to the labeling of the vertices of hyperedge i .

Figure 1 gives a way to visualize Goldreich’s PRG. We interpret our hypergraph H as a bipartite graph B where the input vertices of B represent vertices of H , the output vertices of B represent the hyperedges of H , and edge $(a, b) \in B$ if and only if vertex a is contained in hyperedge b in H . See **Figure 1** for the bipartite graph visualization with an example of the computation of F_H . Our second assumption is the security of Goldreich’s PRG instantiated with the P_5 predicate.

Assumption 4.5. *For every δ , $m = n^{1.5-\delta}$, and $s = \text{poly}(n)$, random Q belonging to set of 5-hypergraphs on n vertices with m hyperedges, and random $x_1, \dots, x_s \in \{0, 1\}^n$, $y_1, \dots, y_s \in \{0, 1\}^m$, $(Q, F_Q(x_1), \dots, F_Q(x_s))$ and (Q, y_1, \dots, y_s) are $o(1)$ -indistinguishable in $n^{O(1)}$ time.*

We now justify **Assumption 4.5**. Notice that it is too much to hope for $\text{negl}(n)$ -indistinguishability in **Assumption 4.5**. To see why, notice that there is a $\Omega(1/n^5)$ chance that the first two hyperedges in Q contain the exact same vertices in the same order, which would cause the first bit of the output of F_Q to always equal the second bit of the output. Such a function F_Q is clearly not a PRG.

The authors of [BKR23] use the weaker assumption that for every δ , $m = n^{1.5-\delta}$, random $Q, x \in \{0, 1\}^n, y \in \{0, 1\}^m$, that $(Q, F_Q(x))$ and (Q, y) are $o(1)$ -indistinguishable in $n^{O(1)}$ time. This differs from **Assumption 4.5** in that it only guarantees $o(1)$ -indistinguishability for one sample from the PRG. However, **Assumption 4.5** is in line with standard assumption about Goldreich’s PRG [LV17, CDM+18].

4.2 The construction

Construction 4.6 (Hyperloop Construction, Hyperloop $[\delta, m, \ell, t]$). *Let δ, m, ℓ, t be efficiently computable functions of the security parameter n .*

- *KeyGen(1^n): Sample H and S as in **Construction 4.2** conditioned on all hyperedges of S_1 being pairwise disjoint. output $(sk = S, pk = H)$.*
- *Encode($1^n, H, 1$): Sample $u \leftarrow \{0, 1\}^n$ and output $F_H(u)$.*
- *Decode($1^n, S_1, x$): Compute $w = \bigoplus_{j \in S_1} x_j$. If $w = 0$, output 1, otherwise output \perp .*

4.3 Robustness

We first review a basic fact about decoding in planted hyperloop graphs when the output is not subjected to errors.

Lemma 4.7 (Claim 1 in [BKR23]). *If H, S come from **Construction 4.2** where the hyperedges of S_1 are disjoint, if we sample x uniformly at random from $\{0, 1\}^m$ and let $y = F_H(x)$, then $w = \bigoplus_{j \in S_1} y_j$ has bias $2^{-\ell}$ towards being 0.*

Proof. We use the notation x_{j_i} to denote the value of the i th vertex of hyperedge j when x is projected onto the vertices of H . Since all vertices in S_1 have degree two, $\bigoplus_{j \in S_1} (x_{j_1} \oplus x_{j_2} \oplus x_{j_3}) = 0$. So, $\bigoplus_{j \in S_1} y_j = \bigoplus_{j \in S_1} (x_{j_1} \oplus x_{j_2} \oplus x_{j_3} \oplus x_{j_4} x_{j_5}) = \bigoplus_{j \in S_1} x_{j_4} x_{j_5}$. Notice that each term $x_{j_4} x_{j_5}$ in the xor is 1 with probability $1/4$. Furthermore, since all hyperedges in S_1 are disjoint, the terms $x_{j_4} x_{j_5}$ are independent for each j . Therefore, the bias of the xor of these biased independent terms, $\bigoplus_{j \in S_1} y_j$, has bias $2^{-\ell}$. \square

We now use this to prove that the output of Goldreich's PRG instantiated with planted hyper-loop graphs is indeed a robust to errors.

Lemma 4.8. *Let δ, m, ℓ, t be the parameters specified in [Assumption 4.3](#) and p be any constant in $[0, 1/2)$. There exists some polynomial $p(n)$ such that for [Construction 4.6](#), for all $d \leq pm$, for all keys $(\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(1^n)$,*

$$\Pr_{\mathcal{E}}[\text{Decode}(1^n, \text{sk}, \mathcal{E}(x)) = 1 : x \leftarrow \text{Encode}(1^n, \text{pk}, 1)] \geq \frac{1}{2} + \frac{1}{p(n)} .$$

Here \mathcal{E} is the d -hypergeometric channel and the randomness is over the randomness of the encoding algorithm and the errors of \mathcal{E} .

Proof. Let $p'(n)$ be any polynomial such that the following holds:

$$\frac{1}{2} \left(1 - 2 \frac{pm}{m - \ell} \right)^\ell \geq \frac{1}{2} + 1/p'(n) .$$

We will show that for all keys, over the randomness in the encoding function and \mathcal{E} , that a codeword from the PRC has a decent chance of being decoded to 1. Fix the key public and private key H, S . Formally, we wish to show that

$$\Pr_{u, \mathcal{E}}[\text{Decode}(1^n, S_1, \mathcal{E}(F_H(u))) = 1] = \frac{1}{2} + \frac{1}{p(n)}$$

Since \mathcal{E} is the d -hypergeometric channel, we can model channel \mathcal{E} as an error vector $e \leftarrow \mathcal{S}_{d,m}$. So the above statement is equivalent to

$$\Pr_{u, e}[\text{Decode}(1^n, S_1, F_H(u) \oplus e) = 1] = \frac{1}{2} + \frac{1}{p(n)}$$

Let us determine the probability that w_i is zero for a fixed H, S . By [Lemma 4.7](#) and [Lemma 2.5](#) respectively,

$$\begin{aligned} \Pr_u[\bigoplus_{j \in S_1} (F_H(u)_j) = 0] &\geq \frac{1}{2} + \frac{1}{n} \\ \Pr_e[\bigoplus_{j \in S_1} e_j = 0] &\geq \frac{1}{2} + \frac{1}{2} \left(1 - 2 \frac{pm}{m - \ell} \right)^\ell \geq \frac{1}{2} + \frac{1}{p'(n)} . \end{aligned}$$

Since $F_H(u_i)$ and e_i are independent, the events $\bigoplus_{j \in S_1} (F_H(u_i)_j) = 0$ and $\bigoplus_{j \in S_1} e_{ij} = 0$ are independent, and therefore, by [Lemma 2.2](#) we have

$$\Pr_{u, e}[w = 0] \geq \frac{1}{2} + \frac{1}{n \cdot p'(n)} . \quad \square$$

4.4 A form of soundness

Lemma 4.9. *Let δ, m, ℓ, t be the parameters specified in [Assumption 4.3](#). For [Construction 4.6](#), for any key pair $(pk, sk) \leftarrow \text{KeyGen}(1^n)$,*

$$\Pr_{x \leftarrow \{0,1\}^n} [\text{Decode}(1^n, sk, x) = 1] = 1/2 .$$

Proof. Recall that decoding in [Construction 4.6](#) outputs \perp if some fixed $O(\log n)$ bits (depending on sk) of x xor to 1. For a random string x , this happens with probability exactly $1/2$. \square

4.5 Pseudorandomness

Lemma 4.10. *Let δ, m, ℓ, t be the parameters specified in [Assumption 4.3](#). Under [Assumption 4.3](#) and [Assumption 4.5](#), the outputs of [Construction 4.6](#) are $o(1)$ -indistinguishable from random by any PPT adversary.*

Proof. Say that the distinguishing algorithm gets $s = \text{poly}(n)$ samples. We will now define several distributions. Let H be the distributions defined in [Construction 4.2](#), H' be the distribution defined for H in [Construction 4.6](#), G be the uniform distribution over 5-hypergraphs with n vertices and m hyperedges, X_1, \dots, X_s all be uniform on $\{0,1\}^n$, and Y_1, \dots, Y_s all be uniform on $\{0,1\}^m$. For any graph g , let $g(x)$ denote the output of Goldreich's PRG instantiated with the P_5 predicate and graph g on input x . By [Fact 2.9](#) and the fact that the KeyGen algorithm in [Construction 4.6](#) rejects a $o(1)$ fraction of keys, the statistical distance between H and H' is $o(1)$. Therefore, $(H, H(X_1), \dots, H(X_s))$ and $(H', H'(X_1), \dots, H'(X_s))$ are distinguishable with advantage at most $o(1)$. By [Assumption 4.3](#) $(H, H(X_1), \dots, H(X_s))$ is distinguishable from $(G, G(X_1), \dots, G(X_s))$ with $o(1)$ advantage. By [Assumption 4.5](#), $(G, G(X_1), \dots, G(X_s))$ is distinguishable from (G, Y_1, \dots, Y_m) with at most $1 - o(1)$ advantage. These three facts along with the hybrid lemma tell us that $(H', H'(X_1), \dots, H'(X_s))$ is distinguishable from (G, Y_1, \dots, Y_s) with advantage at most $o(1)$. \square

4.6 Putting it all together

Theorem 4.11. *Let δ, m, ℓ, t be the parameters specified in [Assumption 4.3](#) and p be a constant in $[0, 1/2)$. Under [Assumption 4.3](#) and [Assumption 4.5](#), [Construction 4.6](#) is a $(1 - \text{negl}(n), 1 - \text{negl}(n), o(1))$ -public-key PRC robust to any p -bounded channel and pseudorandom against all PPT adversaries.*

Proof. Observe that the length of the codewords generated by [Construction 4.6](#) is $\text{poly}(n)$. Furthermore, [Lemma 4.8](#), [Lemma 4.9](#), and [Lemma 4.10](#) give us the remaining necessary preconditions to apply [Lemma 2.18](#) which gives $(1 - \text{negl}(n), 1 - \text{negl}(n), o(1))$ -public-key PRC. \square

The $o(1)$ pseudorandomness in [Theorem 4.11](#) is not ideal but for the purpose of watermarking, seems tolerable. We argue that even $1 - \Omega(1)$ pseudorandomness seems tolerable. In the case of watermarking, indistinguishability is generally to ensure that the quality of model outputs are not degraded. One can therefore sample $j = O(1)$ PRC generators G_1, \dots, G_j . To generate, run the

model j times, and on the i th iteration, watermark using generator G_i . With high probability, one of the j outputs of the model is of non-degraded quality.

To see why, we return to the definition of indistinguishably. For any polynomial time adversary, \mathcal{A} , since the outputs of a random PRC generator are distinguishable from random with $1 - \Omega(1)$ advantage by \mathcal{A} , with high probability, there exists an i such that the distinguishing algorithm \mathcal{A} outputs the same thing when the content is watermarked using generator G_i as it does when the content is not watermarked. This implies that the content watermarked with the PRC G_i is not degraded.

5 The weak planted XOR construction

Christ and Gunn [CG24] gave a scheme which is secure if both the planted XOR assumption and polynomial hardness of LPN with constant noise rate hold ¹. We observe that while polynomial hardness of LPN with constant noise rate is a well believed assumption, the planted XOR assumption is a non-standard and relatively unstudied assumption. Therefore, the conjunction of these two assumptions is quite a strong assumption. It therefore seems plausible a scheme based on a strengthened LPN assumption and a weakened planted XOR assumption is more secure than the one presented in [CG24]. To present such a scheme, we first introduce the weakened planted XOR assumption.

5.1 The assumption

Let us define $\mathcal{D}_0(m, n)$ as the uniform distribution over $\{0, 1\}^{n \times m}$ and we will now define the distribution $\mathcal{D}_1(n, m, t, \varepsilon)$ which corresponds to the distribution of matrices where we strategically implant a low weight vector in the row space.

Construction 5.1 (Generalization of [ASS⁺23]). *We define the distribution $\mathcal{D}_1(n, m, t, \varepsilon)$*

1. Sample $G \leftarrow \{0, 1\}^{n \times m}$,
2. Choose a random tuple $(a_1, \dots, a_t) \subseteq [n]^t$ such that $i \neq j$ implies $a_i \neq a_j$,
3. Let $u = G_{a_1} \oplus \dots \oplus G_{a_{t-1}}$, $v \leftarrow \text{Ber}(m, \varepsilon)$, and update G_{a_t} to $u + v$
4. Output (G, s) where $s \in \{0, 1\}^n$ is the t sparse indicator vector for (a_1, \dots, a_t) .

We are now ready to introduce the (weak) planted XOR assumption.

Assumption 5.2. *For $m, t : \mathbb{N} \rightarrow \mathbb{N}$ and $\varepsilon : \mathbb{N} \rightarrow [0, 1/2]$ which are efficiently computable functions of n , the XOR $_{m,t,\varepsilon}$ assumption states that for every probabilistic polynomial-time adversary \mathcal{A} ,*

$$\left| \Pr_{G \leftarrow \mathcal{D}_0(n,m)} [\mathcal{A}(G) = 1] - \Pr_{(G,s) \leftarrow \mathcal{D}_1(n,m,t,\varepsilon)} [\mathcal{A}(G) = 1] \right| = \text{negl}(n)$$

¹For a particular setting of parameters, it is also secure if LPN with constant noise rate is $2^{O(\sqrt{n})}$ hard

What is referred to as the planted XOR assumption in [CG24] is simply $\text{XOR}_{m,O(\log n),0}$. As one of their major contributions, the authors of [CG24] give a PRC scheme which is secure if (i) [Assumption 5.2](#) with $\varepsilon = 0$, and $m = n^{1-\Omega(1)}$, $t = \Theta(\log n)$ is true, and (ii) constant noise rate LPN is hard. In this section, we show that such a scheme can be based on a more expansive set of assumptions. Informally, we will show that if for any $m = \text{poly}(n)$, $\text{XOR}_{m,\Theta(\log n),O(\log(m)/(m\eta))}$ holds and $\text{LPN}[\eta]$ holds, then pseudorandom codes exist. For concreteness, one may wish to read this section with the parameter regime $\eta = 1/\sqrt{n}$ in mind since $\text{LPN}[1/\sqrt{n}]$ is a well believed assumption and the weakest LPN assumption known to imply public-key cryptography [Ale03].

5.2 Evidence $\text{XOR}_{m,t,\varepsilon}$ is a weaker assumption than $\text{XOR}_{m,t,0}$

Before proceeding with our PRC construction, we give two pieces of evidence that $\text{XOR}_{m,t,\varepsilon}$ is indeed a weaker assumption than $\text{XOR}_{m,t,0}$. The first is a reduction which shows that $\text{XOR}_{m,t,0}$ implies $\text{XOR}_{m,t,\varepsilon}$.

Theorem 5.3. *For any $m, t : \mathbb{N} \rightarrow \mathbb{N}$ and $\varepsilon : \mathbb{N} \rightarrow [0, 1/2]$ which are efficiently computable functions of n , if the $\text{XOR}_{m,t,0}$ assumption holds and $\text{Ber}(n, \varepsilon)$ is efficiently samplable, then the $\text{XOR}_{m,t,\varepsilon}$ assumption holds.*

Proof. We will show the contrapositive statement that if $\text{XOR}_{m,t,\varepsilon}$ does not hold, then $\text{XOR}_{m,t,0}$ does not hold. If the $\text{XOR}_{m,t,\varepsilon}$ assumption does not hold, then there exists a polynomial time distinguisher \mathcal{A} which can distinguish between $\mathcal{D}_0(n, m)$ and $\mathcal{D}_1(n, m, t, \varepsilon)$ with non-negligible advantage $p(n)$. Consider now the distinguisher \mathcal{A}' , which on an input $G \in \mathbb{F}_q^{n \times m}$, samples $i \leftarrow [1, n]$, $v \leftarrow \text{Ber}(m, \varepsilon)$, creates a new matrix $G' \in \mathbb{F}_q^{n \times n}$ where $G' = G$, sets $G'_i \leftarrow G'_i \oplus v$, and then outputs $\mathcal{A}(G')$.

Notice first that \mathcal{A}' is a polynomial time distinguisher since constructing G' and running $\mathcal{A}(G')$ are efficient computations. We now show that \mathcal{A}' can distinguish between $\mathcal{D}_0(m, n)$ and $\mathcal{D}_1(m, n, t, 0)$ with non-negligible advantage. Let us first consider the case when G is sampled from $\mathcal{D}_0(m, n)$. In this case G' is distributed as $\mathcal{D}_0(m, n)$.

Now let us consider the case when G is sampled from $\mathcal{D}_1(m, n, t, 0)$. We now define three families of distributions:

1. For any $s \in \{0, 1\}^n$, let $\mathcal{D}_0^s(m, n)$ be distribution of G when it is sampled uniformly from $\{0, 1\}^{n \times m}$ subject to $s^T G = 0$.
2. For any $s \in \{0, 1\}^n$, let $\mathcal{D}_1^s(m, n)$ denote the distribution of G when it is sampled as follows: sample $v \leftarrow \text{Ber}(n, \varepsilon)$, sample G uniformly from $\{0, 1\}^{n \times m}$ subject to $s^T G = v$. Also denote $\mathcal{D}_1(m, n) = \mathcal{D}_1(m, n, t, \varepsilon)$.
3. Let $\mathcal{D}_2^s(m, n)$ denote the distribution of G when sample it as follows: sample $i \leftarrow [1, n]$, $v \leftarrow \text{Ber}(n, \varepsilon)$, $G \leftarrow \mathcal{D}_0^s(m, n)$, set $G_i \leftarrow G_i \oplus v$. Let $\mathcal{D}_2(m, n)$ denote the distribution of G as follows: Sample $s \leftarrow \mathcal{S}_{t,n}$ and $G \leftarrow \mathcal{D}_2^s(m, n)$.

Notice that $\mathcal{D}_2^s(m, n) = (t/n)\mathcal{D}_1^s(m, n) + (1 - t/n)\mathcal{D}_0^s(m, n)$ since there is a t/n chance that $s_i = 1$,

in which case G is sampled from $\mathcal{D}_1^s(m, n)$. Therefore, the distribution of $\mathcal{D}_2(m, n)$ is

$$\begin{aligned}
\mathcal{D}_2(m, n) &= \sum_{\substack{s \\ |s|=t}} \frac{\mathcal{D}_2^s(m, n)}{\binom{n}{t}} \\
&= \sum_{\substack{s \\ |s|=t}} \left(\frac{t}{n} \frac{\mathcal{D}_1^s(m, n)}{\binom{n}{t}} + \left(1 - \frac{t}{n}\right) \frac{\mathcal{D}_0^s(m, n)}{\binom{n}{t}} \right) \\
&= \frac{t}{n} \sum_{\substack{s \\ |s|=t}} \frac{\mathcal{D}_1^s(m, n)}{\binom{n}{t}} + \left(1 - \frac{t}{n}\right) \sum_{\substack{s \\ |s|=t}} \frac{\mathcal{D}_0^s(m, n)}{\binom{n}{t}} \\
&= \frac{t}{n} \mathcal{D}_1(m, n) + \left(1 - \frac{t}{n}\right) \sum_{\substack{s \\ |s|=t}} \mathcal{D}_0(m, n)
\end{aligned}$$

In our reduction, if $G \leftarrow \mathcal{D}_0(m, n)$, then $G' \sim \mathcal{D}_0(m, n)$, and if $G \leftarrow \mathcal{D}_1(m, n, t, \varepsilon)$, then $G' \sim \mathcal{D}_2(m, n)$. The distinguishing advantage of \mathcal{A} on these two distributions is

$$\begin{aligned}
&\left| \Pr_{x \leftarrow \mathcal{D}_0(m, n)} [\mathcal{A}(x) = 1] - \Pr_{x \leftarrow \mathcal{D}_2(m, n)} [\mathcal{A}(x) = 1] \right| \\
&= \left| \Pr_{x \leftarrow \mathcal{D}_0(m, n)} [\mathcal{A}(x) = 1] - \frac{t}{n} \Pr_{x \leftarrow \mathcal{D}_1(m, n)} [\mathcal{A}(x) = 1] - \left(1 - \frac{t}{n}\right) \Pr_{x \leftarrow \mathcal{D}_0(m, n)} [\mathcal{A}(x) = 1] \right| \\
&= \frac{t}{n} \left| \Pr_{x \leftarrow \mathcal{D}_0(m, n)} [\mathcal{A}(x) = 1] - \Pr_{x \leftarrow \mathcal{D}_1(m, n)} [\mathcal{A}(x) = 1] \right| \\
&= \frac{t}{n} p(n)
\end{aligned}$$

Since $f(n)$ is non-negligible, $(t/n)p(n)$ is non-negligible. Therefore \mathcal{A} distinguishes $\mathcal{D}_0(m, n)$ (the distribution of G' when G comes from $\mathcal{D}_0(m, n)$) and $\mathcal{D}_2(m, n)$ (the distribution of G' when G comes from $\mathcal{D}_1(m, n, t, \varepsilon)$) with non-negligible probability. So \mathcal{A}' is a distinguisher falsifying the $\text{XOR}_{m, t, \varepsilon}$ assumption. \square

Our second piece of evidence that $\text{XOR}_{m, t, \varepsilon}$ is a weaker assumption than $\text{XOR}_{m, t, 0}$ is that $\text{XOR}_{m, t, \varepsilon}$ seems more robust to known attacks than $\text{XOR}_{m, t, 0}$. The first version of [CG24] assumed $\text{XOR}_{\Theta(n), t, 0}$. However, a subsequent version of [ASS⁺23] gave an attack showing $\text{XOR}_{\Theta(n), O(\log n), 0}$ is not true. The attack (Thm 4.26 of [ASS⁺23]) consists of sampling random $m/2 \times m$ submatrices of the input matrix G and then using Gaussian elimination to determine the submatrix contains a sparse subset of rows which xor to zero. The newest version of [CG24] circumvents this problem by setting $m = n^{1-\Omega(1)}$. We note that while $\text{XOR}_{\Theta(n), O(\log n), 0}$ is susceptible to this type of attack, $\text{XOR}_{\Theta(n), O(\log n), \varepsilon}$ is not for reasonable ε (say $\varepsilon = 1/\sqrt{m}$). When attempting this attack against $\text{XOR}_{\Theta(n), O(\log n), 0}$, we can use Gaussian elimination since we were looking for a zero vector in a $m/2$ dimensional subspace. When attempting this attack against $\text{XOR}_{\Theta(n), O(\log n), \varepsilon}$, we must find a low weight vector in a $m/2$ dimensional subspace. This problem is the average case version of the problem finding a planted low weight codeword v in a linear code, a problem which is generally believed to be intractable.

5.3 The construction

Construction 5.4 (Weak sparse xor construction, $\text{weakXOR}[m, t, \varepsilon, \eta]$). Let m, t, ε, η be efficiently computable functions of the security parameter n

- $\text{KeyGen}(1^n)$: Sample (G, s) from $\mathcal{D}_1(n, m, t, \varepsilon)$. Output $(sk = s, pk = G)$.
- $\text{Encode}(1^n, G)$: Sample $u \leftarrow \text{Ber}(m, \eta)$, $e \leftarrow \text{Ber}(n, \eta)$. Output $Gu + e$.
- $\text{Decode}(1^n, s, x)$: If $s^T x = 0$, output 1. Otherwise, output \perp .

5.4 Robustness

Lemma 5.5. Let $m = \text{poly}(n)$, $\eta = o(1)$, $t = O(\log n)$, $\varepsilon = O(\log(m)/(\eta m))$, and p be any constant in $[0, 1/2)$. There exists a polynomial $p(n)$ such that for **Construction 5.4**, for any $d \leq pn$, for a $1 - \text{negl}(n)$ fraction of keys $(pk, sk) \leftarrow \text{KeyGen}(1^n)$,

$$\Pr_{\mathcal{E}}[\text{Decode}(1^n, sk, \mathcal{E}(x)) = 1 : x \leftarrow \text{Encode}(1^n, pk, 1)] \geq \frac{1}{2} + \frac{1}{p(n)}$$

where \mathcal{E} is the d -hypergeometric channel. the randomness is over the randomness of the encoding algorithm and the errors of \mathcal{E} .

Proof. Let $p(n) = p'(n) \cdot p''(n) \cdot p'''(n)$ where $p'(n)$, $p''(n)$, and $p'''(n)$ are polynomials we will choose later. Consider the key sampling procedure and let $s^T G = v$. Notice that by **Lemma 2.3**, there is a $e^{-\Omega(\varepsilon m)}$ chance that $|v| \geq 1.5\varepsilon m$. As long as $\varepsilon = \omega(\log(m)/m)$, this means there is a $1 - \text{negl}(n)$ chance (over the key sampling procedure) that $|v| \leq 1.5\varepsilon m$. Let $p'(n)$ be a polynomial so that $(1 - 2\eta)^{1.5\varepsilon m} \geq 1/p'(n)$. We will assume for the remainder of the proof that $(1 - 2\eta)^{|v|} \geq 1/n^c$ since this happens for a $1 - \text{negl}(n)$ fraction of keys.

Fix the keys. We must show that over the randomness encoding function and the channel, that a codeword from the PRC has a reasonable chance of being decoded to one. Formally, we wish to show that

$$\Pr_{u, e, \mathcal{E}}[\text{Decode}(1^n, s, \mathcal{E}(Gu + e)) = 1] = \frac{1}{2} + \frac{1}{p(n)}$$

Since \mathcal{E} is the d -hypergeometric channel, we can model the error from the channel as an error vector $e' \leftarrow \mathcal{S}_{d, n}$. We then need to prove

$$\Pr_{u, e, e'}[\text{Decode}(1^n, s, Gu + e + e') = 1] = \frac{1}{2} + \frac{1}{p(n)}$$

By the definition of our decoding function, the above probability is equal to

$$\Pr_{u, e, e'}[s^T(Gu + e + e') = 0] = \Pr_{u, e, e'}[(s^T G)u + s^T e + s^T e' = 0]$$

Since $(s^T G)u = v^T u$ is the xor of the coordinates of u on which v is one, $v^T u$ is the xor of $|v|$ i.i.d $\text{Ber}(\eta)$ random variables. Therefore, by **Lemma 2.2**, $(s^T G)u = vu$ has probability $1/2 + (1 - 2\eta)^{|v|} = 1/2 + 1/p'(n)$ of being zero. Similarly, let $p''(n)$ be a polynomial (which we know

exists by [Lemma 2.2](#)) so that $s^T e$ has at least a $1/2 + 1/p''(n)$ chance of being 0. Let $p'''(n)$ be a sufficiently large constant so that $1/2 + (1/2)(pn/(n-t))^t \geq 1/2 + 1/p'''(n)$. Finally, by [Lemma 2.5](#), $s^T e'$ also has at least $1/2 + (1/2)(pn/(n-t))^t \geq 1/2 + 1/p'''(n)$ chance of being 0. Since these events are all independent $s^T Gu + s^T e + s^T e'$ has probability at least $1/2 + 1/p(n)$ of being 0 by [Lemma 2.2](#). \square

5.5 A form of soundness

Lemma 5.6. *For any m, t, ε, η , $k = \text{poly}(n)$, for [Construction 5.4](#), for all key pairs $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(1^n)$, we have*

$$\Pr_{x \leftarrow \{0,1\}^n} [\text{Decode}(1^n, \text{sk}, x) = \perp] = \frac{1}{2}$$

Proof. Recall that decoding in [Construction 5.4](#) outputs \perp if some fixed $O(\log n)$ bits (depending on sk) of x xor to 1. For a random string x , this happens with probability exactly $1/2$. \square

5.6 Pseudorandomness

Lemma 5.7. *For any efficiently computable $m = \text{poly}(n), t, \varepsilon, \eta$, if $\text{XOR}_{m,t,\varepsilon}$ holds, and $\text{LPN}[\eta]$ holds, then $\text{weakXOR}[m, t, \varepsilon, \eta]$ is pseudorandom.*

Proof. Let G be distributed uniformly over $\{0, 1\}^{n \times m}$, G' be distributed according to $\mathcal{D}_1(m, n, t, \varepsilon)$, $q = \text{poly}(n)$, x_1, \dots, x_q be distributed as $\text{Ber}(m, \eta)$, and e_1, \dots, e_q be distributed as $\text{Ber}(m, \eta)$. The output distribution of $\text{weakXOR}[m, t, \varepsilon, \eta]$ is $G'x_1 + e_1, \dots, G'x_q + e_q$. By the $\text{XOR}_{m,t,\varepsilon}$ assumption, that distribution is indistinguishable from $Gx_1 + e_1, \dots, Gx_q + e_q$. The $\text{LPN}[\eta]$ assumption implies $\text{LPN}'[\eta]$, which implies that $Gx_1 + e_1, \dots, Gx_q + e_q$ is indistinguishable from random. By the hybrid lemma, $G'x_1 + e_1, \dots, G'x_s + e_s$ is also indistinguishable from random. \square

Remark 5.8. Technically, we require something weaker than $\text{LPN}[\eta]$ to hold for our proof of pseudorandomness. We need only that LPN is secure when the distribution of the secret comes from $\text{Ber}(m, \eta)$ and the error comes from a distribution $\text{Ber}(n, p)$ for any constant $p < 1/2$. However, since the LPN assumption is typically stated solely in terms of the error rate and $\text{LPN}[\eta]$ is sufficient for this construction, we choose to state our results as being based on the (possibly stronger than necessary) $\text{LPN}[\eta]$ assumption.

5.7 Putting it all together

Theorem 5.9. *For efficiently computable $m = \text{poly}(n), t = O(\log n), \eta = o(1), \varepsilon = O(\log(m)/(\eta m))$ which are functions of n and constant $p \in [0, 1/2)$, if $\text{XOR}_{m,t,\varepsilon}$ holds and $\text{LPN}[\eta]$ holds, then there exists a $(1 - \text{negl}(n), 1 - \text{negl}(n), \text{negl}(n))$ -public-key PRC which is robust to all p -bounded channels and pseudorandom against all PPT adversaries.*

Proof. Observe that the length of the codewords generated by [Construction 5.4](#) is $\text{poly}(n)$. Furthermore, [Lemma 5.5](#), [Lemma 5.6](#), and [Lemma 5.7](#) give us the remaining necessary preconditions to apply [Lemma 2.18](#) which gives $(1 - \text{negl}(n), 1 - \text{negl}(n), \text{negl}(n))$ -public-key PRC. \square

6 PRCs for space-bounded adversaries

We now present a zero-bit PRC scheme based on the time-space hardness of the learning parity with noise problem which is robust $\text{BSC}(p)$ for any constant $p < 1/2$. The pseudorandomness of this construction is *unconditional* and not based on cryptographic assumptions. The celebrated work of [Raz18] showed that the learning parity without noise problem requires either a superpolynomial number of samples or $\Omega(n^2)$ memory. Follow-up work [KRT17] and [GKLR21] expanded this work to the cases where the secret is sparse and the case where the samples are noisy. We will begin by reviewing the relevant definitions and results.

Definition 6.1. *The learning sparse parities problem with density ℓ and error rate ε is defined as follows: The secret vector s is sampled uniformly at random from $\mathcal{S}_{\ell,n}$. An algorithm \mathcal{A} is given samples $(a, a \cdot s + e)$ where $a \leftarrow \{0, 1\}^n, e \leftarrow \text{Ber}(1/2 - \varepsilon)$. We say \mathcal{A} succeeds if it successfully outputs s .*

Definition 6.2. *We say that a distribution of bits X_1, \dots, X_n is next-bit unpredictable for a class of adversaries \mathcal{C} if for all $\mathcal{A} \in \mathcal{C}$ and all $i \in [1, n]$, there exists a negligible function $\varepsilon(n)$ such that*

$$\Pr[\mathcal{A}(1^n, X_1, \dots, X_{i-1}) = X_i] \leq \frac{1}{2} + \varepsilon(n)$$

Lemma 6.3. *Let $q = \text{poly}(n)$ and $\varepsilon = o(1)$. The distribution $a_1, a_1 \cdot s + e_1, \dots, a_q, a_q \cdot s + e_q$ where $s \leftarrow \mathcal{S}_{\Theta(\log n), n}$ and $a_i \leftarrow \{0, 1\}^n, e_i \leftarrow \text{Ber}(1/2 - \varepsilon)$ for all $i \in [1, q]$ is next-bit unpredictable for PPT algorithms with $O(n \log^{0.99}(n)/\varepsilon)$ space.*

See [Section 7](#) for a derivation of [Lemma 6.3](#).

6.1 Construction

Construction 6.4 (small space resilient construction, $\text{SSR}[\ell, \varepsilon, k, \delta]$). *Let ℓ, ε, k' be efficiently computable functions of the security parameter n and δ be a constant.*

- *KeyGen(1^n):* Sample $s_1, \dots, s_{k'} \leftarrow \mathcal{S}_{\ell,n}$ and output $sk = (s_1, \dots, s_{k'})$.
- *Encode($1^n, (s_1, \dots, s_{k'}), 1$):* Sample $a \leftarrow \{0, 1\}^n, e_1, \dots, e_{k'} \leftarrow \text{Ber}(1/2 - \varepsilon)$, output

$$a \parallel a \cdot s_1 + e_1 \parallel \dots \parallel a \cdot s_{k'} + e_{k'} .$$

- *Decode($1^n, (s_1, \dots, s_{k'}), x$):* Reinterpret $x \in \{0, 1\}^{n+k'}$ as $\tilde{a} \parallel \tilde{b}_1 \parallel \dots \parallel \tilde{b}_{k'}$ where $\tilde{a} \in \{0, 1\}^n$ and $\tilde{b}_i \in \{0, 1\}$ for all $i \in [1, k']$. If \tilde{a} is not $1/(2n^{0.4})$ balanced, output \perp . Otherwise, let w_i be one if and only if $\tilde{a} \cdot s_i = \tilde{b}_i$. If $\sum_{i=1}^{k'} w_i \geq k'/2 + n^\delta \sqrt{k'}$ output 1 and otherwise output \perp .

6.2 Robustness

Say that the decoder receives a string $x = \tilde{a} \parallel \tilde{b}_1 \parallel \dots \parallel \tilde{b}_{k'}$. Intuitively, for every i such that $\tilde{a} \cdot s_i = \tilde{b}_i$, the decoder gains more confidence that x is a codeword. However, on first inspection, it seems plausible one could flip a just a few of the first n bits of a codeword (turn a into \tilde{a}) to ensure there

would exist very few $i \in [k']$ such that $\tilde{a} \cdot s_i = \tilde{b}_i$. The existence of such an attack could potentially imply that the code of [Construction 6.4](#) is not particularly robust to errors. We will show that such an attack does not affect robustness due to the sparsity of the s_i . In order to do so, we first review a version of the Chernoff bound for weakly dependent random variables.

Definition 6.5 ([GLSS12]). *A family $Y_1, \dots, Y_{k'}$ of random variables is read- d if there exists a sequence X_1, \dots, X_n of independent variables, and a sequence $S_1, \dots, S_{k'}$ of subsets of $[n]$ such that*

1. *Each Y_i is a function of $(X_j : j \in S_i)$, and*
2. *No element of $[n]$ appears in more than d of the S_i 's.*

Lemma 6.6 ([GLSS12]). *Let $Y_1, \dots, Y_{k'}$ be a family of read- d indicator random variables with $\Pr[Y_i = 1] = p_i$ and let p be the average of $p_1, \dots, p_{k'}$. Then for any $\varepsilon > 0$, the probabilities*

$$\Pr[Y_1 + \dots + Y_{k'} \geq (p + \varepsilon)k'] \quad \text{and} \quad \Pr[Y_1 + \dots + Y_{k'} \leq (p - \varepsilon)k']$$

are both at most $e^{-2\varepsilon^2 k'/d}$

Lemma 6.7. *Let $\ell \leq O(\log n)$, $d = \omega(\log n)$, and $k' \leq n$. Consider $S = \{S_1, \dots, S_{k'}\}$ where each S_i is drawn uniformly at random from $\binom{[n]}{\ell}$. Some element $t \in [n]$ occurs in d elements of S with probability $\text{negl}(n)$.*

Proof. Let T_t denote the event where t occurs in at least d elements of S . Since t occurs in each element of S independently with probability ℓ/n , the probability that it appears in at least d elements of S is the probability that a random variable distributed as $\text{Bin}(k', \ell/n)$ is at least d . Since $k' \leq n$ and $d = \omega(\log n)$, this is at most the probability that a random variable distributed as $\text{Bin}(n, \ell/n)$ is at least $\omega(\log n)$. By [Lemma 2.3](#), this probability is at most $\text{negl}(n)$. Union bounding over all $t \in [n]$, we see that probability that there exists some element $t \in [n]$ occurring in more than d sets is at most $n \cdot \text{negl}(n) = \text{negl}(n)$. \square

Lemma 6.8. *Let ε be some function of n , p be a constant in $[0, 1/2)$, $\delta > 0$, and $k' = (2n^{2\delta}/\varepsilon)^2$. There exists a constant $c > 0$ such that for $\ell = c \log(n)$, $\text{SSR}[\ell, \varepsilon, k', \delta]$ is robust to $\text{BSC}(p)$ with probability $1 - \text{negl}(n)$.*

Proof. The probability that a codeword is decoded to 1 correctly is equal to the probability that the following experiment succeeds. We sample $s_1, \dots, s_{k'} \leftarrow \mathcal{S}_{\ell, n}$, $e_1, \dots, e_{k'} \sim \text{Ber}(1/2 - \varepsilon)$, $a \leftarrow \{0, 1\}^n$, and e' from $\text{Ber}(n + k', p)$. Let $\tilde{a} = a \oplus e'_{[1, n]}$, $\tilde{b}_i = a \cdot s_i + e_i + e'_{n+i}$ for every $i \in [1, k']$, and $w_i = 1$ if and only if $\tilde{a} \cdot s_i = \tilde{b}_i$. The experiment succeeds if $\sum_{i=1}^{k'} w_i \geq k'/2 + n^\delta \sqrt{k'}$.

By [Lemma 6.7](#), we can fix $S = \{S_1, \dots, S_{k'}\}$ and assume that no element $t \in [n]$ occurs in more than $n^\delta/2 = \omega(\log n)$ elements of S since this is true with $1 - \text{negl}(n)$ probability.

We see that each w_i is a function of $\{a_r : r \in S_j\} \cup \{e'_r : r \in S_j\} \cup \{e_i, e'_{n+i}\}$. Since we assumed no element $t \in [n]$ occurs in more than $n^\delta/2$ of the sets S_j , we see that no $w_i, w_{i'}$ where $i \neq i'$ share more than n^δ random variables on which they are dependent. Therefore, w_i are read- n^δ random variables. We will now compute the expectation of w_i :

$$\mathbb{E}[w_i] = \Pr[\tilde{a} \cdot s_i = \tilde{b}_i]$$

$$\begin{aligned}
&= \Pr[(a + e'_{[1,n]}) \cdot s_i = (a \cdot s_i) + e_i + e'_{n+i}] \\
&= \Pr[e'_{[1,n]} \cdot s_i = e_i + e'_{n+i}].
\end{aligned}$$

Recall $|s_i| = O(\log n)$, and by symmetry, we can assume without loss of generality that s_i is a series of ones followed by a series of zeros. So the above probability expression is equal to

$$\begin{aligned}
&= \Pr[e'_{[1, c \log(n)]} + e_i + e'_{n+i} = 0] \\
&= \frac{1}{2} \left(1 + (1 - 2p)^{c \log(n)+1} \left(1 - 2 \left(\frac{1}{2} - \varepsilon \right) \right) \right) \\
&= \frac{1}{2} \left(1 + 2\varepsilon(1 - 2p)^{c \log(n)+1} \right)
\end{aligned}$$

where the second equality follows from [Lemma 2.2](#). We can set c to be a sufficiently small constant such that $(1 - 2p)^{c \log(n)+1} \geq 1/n^\delta$ so that the above is at least $\frac{1}{2} + \varepsilon/n^\delta$.

Now that we know the expected value for each w_i , we can use the Chernoff bound for variables with bounded dependence. For sufficiently large n , there are k' such read- n^δ variables and each has probability at least $1/2 + \varepsilon/n^\delta$ of being 1. The probability that the decoding algorithm decodes to \perp is

$$\begin{aligned}
\Pr \left[w_1 + \dots + w_{k'} \leq k'/2 + n^\delta \sqrt{k'} \right] &= \Pr \left[\sum_{i=1}^{k'} w_i \leq (1/2 + \varepsilon/n^\delta - \varepsilon/n^\delta + n^\delta/\sqrt{k'})k' \right] \\
&\leq e^{-2(-\varepsilon/n^\delta + n^\delta/\sqrt{k'})^2 k'/n^\delta} \\
&= e^{-\Omega(\varepsilon/n^\delta)^2 k'/n^\delta} \\
&= e^{-\Omega(\varepsilon^2/n^{2\delta})k'/n^\delta} \\
&= e^{-\Omega(\varepsilon^2/n^{2\delta}) \cdot \Omega(n^{4\delta}/\varepsilon^2)/n^\delta} \\
&= e^{-\Omega(n^{2\delta})/n^\delta} \\
&= \text{negl}(n)
\end{aligned}$$

where the second inequality follows from [Lemma 6.6](#) and the third follows by assumption on the value of k' . Therefore, there is a $\text{negl}(n)$ chance that codeword is decoded to \perp . \square

6.3 Soundness

On first inspection, it may seem strange that we output \perp when trying to decode strings where \tilde{a} is not balanced. This is to ensure soundness. To see why this exit condition is necessary, consider what happens when the codeword is the string of all zeros. [Construction 6.4](#) would certainly decode this codeword to 0 regardless of what \mathbf{sk} is. The requirement that \tilde{a} eliminates the possibility of such edge cases. We will now show the soundness of our zero bit encryption scheme by showing that any fixed $x \in \{0, 1\}^{n+k'}$ decodes to \perp with high probability.

Lemma 6.9. *Let $a \in \{0, 1\}^n$ be a $1/n^{0.4}$ -biased string, $b \in \{0, 1\}$, c be an arbitrary constant and s be drawn uniformly at random from $\mathcal{S}_{c \log(n), n}$.*

$$\Pr_s [a \cdot s = b] \leq 1/2 + \text{negl}(n)$$

Proof. Let $r = |\{i : a_i = 1\}|$ and notice $n/2 - n^{0.6} \leq r \leq n/2 + n^{0.6}$ since a is σ -balanced. The probability that $a \cdot s = 0$ is the probability $X \sim \text{Hyp}(n, r, c \log(n))$ is even. Since $1/2 - O(1/n^{0.1}) \leq (r - c \log(n))/n \leq 1/2 + O(1/n^{0.1})$ and $1/2 - O(1/n^{0.1}) \leq r/(n - c \log n) \leq 1/2 + O(1/n^{0.1})$, by [Corollary 2.6](#), the probability that $X \sim \text{Hyp}(n, r, c \log(n))$ is even is at most $1/2 + O(1/n^{0.1})^{c \log(n)} = 1/2 + \text{negl}(n)$. A similar argument shows that the probability that $a \cdot s = 1$ is at most $1/2 + \text{negl}(n)$. \square

Lemma 6.10. *Let $0 \leq \delta \leq 1/100$ be constant, ϵ be any function of n , $k' \geq n^\delta$ be poly(n), and $\ell = O(\log n)$. For any fixed $x \in \{0, 1\}^{n+k'}$, in the $\text{SRR}[\ell, \epsilon, k', \delta]$ scheme,*

$$\Pr_{\text{sk}} [\text{Decode}(\text{sk}, x) = \perp] \geq 1 - \text{negl}(n) .$$

Proof. Let us reanalyze x as $a||b_1|| \dots ||b_{k'}|$ where $a \in \{0, 1\}^n$ and $b_i \in \{0, 1\}$ for $i \in [1, k']$. Consider the set $G = \{j : a \cdot s_j = b_j\}$. Recall that for x to not decode to \perp , we need $|G| \geq k'/2 + n^\delta \sqrt{k'}$. By [Lemma 6.9](#), each i is in G independently with probability $1/2 + \text{negl}(n)$. So the mean value of $|G|$ is $k'/2 + \text{negl}(n)k'$. Therefore

$$\begin{aligned} \Pr_{\text{sk}} [\text{Decode}(\text{sk}, x) \neq \perp] &= \Pr_{\text{sk}} \left[|G| \geq \frac{k'}{2} + n^\delta \sqrt{k'} \right] \\ &\leq \Pr_{\text{sk}} \left[|G| \geq \left(1 + \frac{n^{\delta/100}}{\sqrt{k'}} \right) \left(\frac{k'}{2} + \text{negl}(n) \right) \right] \\ &= \text{negl}(n) \end{aligned}$$

where the second to last inequality is true for sufficiently large n and the last inequality follows from [Lemma 2.3](#). \square

6.4 Pseudorandomness

We will show pseudorandomness of [Construction 6.4](#) by first showing that any polynomial number of codewords is next-bit unpredictable for a polynomial time, space-bounded adversary. [Lemma 6.3](#) shows that sparse parity learning examples $a||a \cdot s + e$ are next bit unpredictable. In this case, a is random and one pseudorandom bit is output per freshly sampled a . However, in [Construction 6.4](#), the samples are of the form $a||a \cdot s_1 + e_1|| \dots ||a \cdot s_{k'} + e_{k'}|$. In this case, a is random and multiple pseudorandom bits are output per freshly sampled a . Fortunately, next-bit unpredictability of samples of the form $a||a \cdot s + e$ implies next-bit unpredictability of samples of the form $a||a \cdot s_1 + e_1|| \dots ||a \cdot s_{k'} + e_{k'}|$.

Lemma 6.11. *Let $0 \leq \delta \leq 1/100$ be a constant, $\epsilon = 1/\text{poly}(n)$, $\log(\epsilon) \in \mathbb{Z}$, $q = \text{poly}(n)$, $k' = \text{poly}(n)$, and $\ell = \Theta(\log n)$. Let Enc be the encoding function of $\text{SSR}[\ell, \epsilon, k', \delta]$. Consider the distribution induced by $\text{sk} \leftarrow \text{KeyGen}(1^n)$ and $X_1, \dots, X_q \leftarrow \text{Enc}(1^n, \text{sk}, 1)$ where $X_i \in \{0, 1\}^{n+k'}$ for all $i \in [1, q]$. No PPT, $O(n \log^{0.99}(n)/\epsilon)$ space adversary acts as a next bit predictor for X_1, \dots, X_q .*

Proof. Consider the following distribution on $q(n+1)$ bits: $Y = (a_1, a_1 \cdot s + e_1, \dots, a_q, a_q \cdot s + e_q)$ where $s \leftarrow \mathcal{S}_{\ell, n}$ and $a_i \leftarrow \{0, 1\}^n, e_i \leftarrow \text{Ber}(1/2 - \epsilon)$. Say for the sake of contradiction that there exists an algorithm a polynomial time, $O(n \log^{0.99}(n)/\epsilon)$ space adversary \mathcal{A} and a series of indices $\{i_n\}_{n \in \mathbb{N}}$ such that \mathcal{A} could predict bit i_n of $(X_1, \dots, X_q) \in \{0, 1\}^{q(n+k')}$ with non-negligible

probability. We will use \mathcal{A} to construct an algorithm \mathcal{A}' which acts as a next bit predictor for Y by predicting bit $\{i'_n\}_{n \in \mathbb{N}}$ of Y with non-negligible probability.

Bit i_n cannot be a truly random bit belonging to a newly sampled a since then it would not be predictable. Therefore, bit i_n corresponds to $(a_g \cdot s_j + e_{g,j})$ for some $g \in [1, q], j \in [1, k']$. In words, i_n is the j th parity bit from codeword g . We now construct \mathcal{A}' that will predict bit $i'_n = j(n+1)$ of Y . \mathcal{A}' begins by sampling each $s_1, \dots, s_{k'}$ *excluding* s_j from $\mathcal{S}_{\Theta(\log n), n}$. \mathcal{A}' then simulates \mathcal{A} .

Recall $Y = (a_1, a_1 \cdot s + e_1, \dots, a_q, a_q \cdot s + e_q)$ and \mathcal{A}' wishes to predict $j(n+1)$ of Y . \mathcal{A}' samples $e_{u,v} \leftarrow \text{Ber}(1/2 - \varepsilon)$ for all $u \in [q], v \in [1, k']$. Let $t_i \in \{0, 1\}^{n+k'}$ be $(a_i, a_i \cdot s_1 + e_{i,1}, \dots, a_i \cdot s_{j-1} + e_{i,j-1}, a_i \cdot s + e_i, a_i \cdot s_{j+1} + e_{i,j+1}, \dots, a_i \cdot s_{k'} + e_{i,k'})$. \mathcal{A}' feeds the first $i_n - 1$ bits of (t_1, \dots, t_q) to \mathcal{A} , which then outputs its prediction b . \mathcal{A}' outputs b .

We now confirm that \mathcal{A}' is a polynomial time, $O(n \log^{0.99}(n)/\varepsilon)$ space algorithm. The sampling of any $e \leftarrow \text{Ber}(1/2 - \varepsilon)$ requires some care. If $\varepsilon = 1/2^x$ for some $x = n^{O(1)}$, we can use x bits to sample an integer y uniformly at random from $[1, 2^x]$. If $y \in [1, 2^{x-1} + 1]$, we set $e = 0$ and otherwise set $e = 1$. This sampling procedure results in $e \sim \text{Ber}(1/2 - \varepsilon)$. The rest of the computation still clearly proceeds in $\text{poly}(n)$ time. Furthermore, \mathcal{A}' only needs $k' \cdot \log(\binom{n}{\log(n)}) = k' \cdot O(\log^2(n)) = O(n)$ auxiliary space to store and compute with $(s_1, \dots, s_{k'})$. Therefore, \mathcal{A}' is a $\text{poly}(n)$ time, $O(n \log^{0.99}(n)/\varepsilon)$ space algorithm.

Since (t_1, \dots, t_q) has the exact same distribution as (X_1, \dots, X_q) , it should be clear that \mathcal{A} predicts bit i_n of (t_1, \dots, t_q) non-negligible probability. Since by construction, that bit corresponds exactly to bit $j(n+1)$ of Y , we see that \mathcal{A}' predicts bit $j(n+1)$ of Y with non-negligible probability.

Therefore, \mathcal{A}' is a $\text{poly}(n)$ time, $O(n \log^{0.99}(n)/\varepsilon)$ space algorithm which predicts bit i'_n of the output of Y . This contradicts [Lemma 6.3](#). \square

Lemma 6.12. *Let $0 \leq \delta \leq 1/100$ be a constant, $\varepsilon = 1/\text{poly}(n)$, $\log(\varepsilon) \in \mathbb{Z}$, $q = \text{poly}(n)$, $k' = \text{poly}(n)$, and $\ell = \Theta(\log n)$. The scheme $\text{SSR}[\ell, \varepsilon, k', \delta]$ is pseudorandom against $O(n \log^{0.99}(n)/\varepsilon)$ space, $\text{poly}(n)$ time adversaries.*

Proof. Follows from [Lemma 6.11](#) and observing that the standard hybrid argument showing that next bit unpredictability implies pseudorandomness [[Gol00](#)] applies even for space-bounded adversaries. \square

6.5 Putting it all together

Theorem 6.13. *Let $0 \leq \delta \leq 1/100$ be a constant, $\varepsilon = 1/\text{poly}(n)$, $k' = (2n^{2\delta}/\varepsilon)^2$, and p be a constant in $[0, 1/2)$. There exists a constant $c > 0$ such that $\text{SSR}[c \log(n), \varepsilon, k, k', \delta]$ is a zero-bit secret-key PRC which*

- has output length $n + k'$
- is robust to $\text{BSC}(p)$
- has key size $O(k' \cdot \log^2(n))$
- pseudorandom against probabilistic polynomial time, $O(n \log^{0.99}(n)/\varepsilon)$ space adversaries.

Theorem 6.13 shows that **Construction 6.4** can have quite small key sizes at the expense of being pseudorandom against adversaries with smaller space. We now instantiate the parameter regime we believe to be the most useful.

Corollary 6.14. *Let $0 \leq \delta \leq 1/100$ be a constant and p be a constant in $[0, 1/2)$. There exists a constant $c > 0$ such that $\text{SSR}[c \log(n), \varepsilon, k, k', \delta]$ is a zero-bit secret-key PRC which*

- has output length $O(n)$
- is robust to $\text{BSC}(p)$
- has key size $O(n)$
- pseudorandom against probabilistic polynomial time, $O(n^{1.5-2\delta}/\log^{0.01}(n))$ space adversaries.

Proof. Follows by setting $\varepsilon = \log(n)n^{-1/2+2\delta}$ in **Theorem 6.13**. □

Therefore, we have shown zero bit PRCs with $O(n)$ length which are *unconditionally* pseudorandom against poly(n) time, $O(n^{1.5-\delta})$ space (for any constant $\delta > 0$) adversaries. It is natural to ask if this leads to multi-bit PRCs. The construction of multi-bit PRCs with rate $1/n$ (construction 3 of [CG24]) also works in the space-bounded setting but has codeword length $O(kn)$ when encoding k bits. This would let us build k -bit PRCs with codeword length $O(kn)$ which are pseudorandom against PPT, $O(n^{1.5-\delta})$ space adversaries. However, that construction has the undesirable property that it narrows the gap between the space of the adversary and the space of the encoding algorithm, thereby making the scheme less secure. It would be interesting to build constant rate PRCs which are unconditionally pseudorandom against PPT, space-bounded adversaries.

7 Perspectives

Here we review some of the design decisions we have made in our constructions.

In **Section 6**, we prove robustness to the binary symmetric channel rather than p -bounded channels (we assume p is a constant in $[0, 1/2)$). One may ask whether it is possible to prove robustness to all p -bounded channels rather than just the binary symmetric channel. To show robustness to p -bounded channels, one could choose to apply a similar type of reduction as given in **Lemma 2.18** by including in the secret key a shift z and a permutation π . This reduces showing robustness against p -bounded channels to showing robustness against d -hypergeometric channels for all $d \leq pn$, which is very similar to the binary symmetric channel. Since the robustness probability only goes up as p goes down in **Lemma 6.8**, there exists a function $u(n) = \text{negl}(n)$ such that for all $d \in [1, pn]$, **Construction 6.4** is robust to $\text{BSC}(d/n)$ with probability $1 - u(n)$. This implies **Construction 6.4** is robust to any d -hypergeometric channel for $d \leq pn$ with probability $O(n)u(n) = \text{negl}(n)$. However, such a reduction incurs an additive $O(n \log n)$ factor in the key size since π is $O(n \log n)$ bits. In the space-bounded setting, having small keys is particularly important, so we have chosen to focus on the standard setting of the binary symmetric channel, which allows for remarkably small key sizes. However, it should not be hard to formalize the argument for p -bounded channels.

This work focuses on the theoretical aspects of PRCs but one can also ask if [Section 4](#) and [Section 5](#) are practical for watermarking LLM text. Unfortunately this seems unlikely. The problem is that if we set the security parameter $n = 128$ (a reasonable security parameter), the application of the [Lemma 2.18](#), which allows us to construct a PRC from a scheme where there is only a small advantage in distinguishing codewords from random words, requires us to concatenate many codewords together, which may result in a code with a length of $\text{poly}(n)$ for some very large polynomial. This is too long to be practical. Fundamentally, [Lemma 2.18](#) allows us to amplify robustness by concatenating t codewords of length n to form a string x of length tn . Every n bit block of $y = \mathcal{E}(x)$ that is decoded to 1 rather than \perp gives us more certainty that y is a corrupted codeword.

There are, however, other ways to amplify our confidence. For example, each codeword of length n can contain multiple checks. In [Construction 5.4](#), (for simplicity consider the $\epsilon = 0$ regime) we sample G uniformly at random subject to $s^T G = 0^m$ and then check if y is a corrupted codeword by checking if $s^T y = 0$. This gives us low confidence that y is a corrupted codeword, so we apply [Lemma 2.18](#). However, imagine we had s_1, \dots, s_τ and sampled G uniformly at random subject to the constraints that $s_i^T G = 0^m$ for all $i \in [1, \tau]$. Then to check if y is a corrupted codeword, we check how many $i \in [1, \tau]$ there were such that $s_i^T y = 0$, and the more there were, the more confidence that we could have that y is a corrupted codeword. This is the approach advocated by [\[CG24\]](#).

Similarly, in [Section 4](#), we implant $\text{poly}(n)$ hyperloops but one use one for decoding (by checking if $\bigoplus_{j \in S_1} y_j = 0$) and then amplify our success probability using [Lemma 2.18](#). From a theoretical perspective, the polynomial size blowup in the length of the code incurred by [Lemma 2.18](#) does not matter. However, from a practical perspective, the correct approach would check how many $i \in [1, t]$ there are such that $\bigoplus_{j \in S_i} y_j = 0$. In both cases, adding more structure in the encoding/decoding stages means that the decoder knows with greater certainty if a word is a codeword, without incurring a large blowup in codeword length.

References

- [ABW10] Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, STOC '10, page 171–180, New York, NY, USA, 2010. Association for Computing Machinery. [5](#)
- [ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, pages 595–618, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. [8](#)
- [Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *FOCS*, pages 298–307, 2003. [19](#)
- [Arr22] Scott Arronson. My ai safety lecture for ut effective altruism, 2022. [12](#)

- [ASS⁺23] Shweta Agrawal, Sagnik Saha, Nikolaj Ignatieff Schwartzbach, Akhil Vanukuri, and Prashant Nalini Vasudevan. k -SUM in the sparse regime. Cryptology ePrint Archive, Paper 2023/488, 2023. <https://eprint.iacr.org/2023/488>. 3, 18, 20
- [BKR23] Andrej Bogdanov, Pravesh Kothari, and Alon Rosen. Public-key encryption, local pseudorandom generators, and the low-degree method. Cryptology ePrint Archive, Paper 2023/1049, 2023. <https://eprint.iacr.org/2023/1049>. i, 3, 5, 11, 13, 14, 15
- [CDM⁺18] Geoffroy Couteau, Aurélien Dupin, Pierrick Meaux, Mélissa Rossi, and Yann Rotella. *On the Concrete Security of Goldreich’s Pseudorandom Generator: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part II*, pages 96–124. 01 2018. 15
- [CG24] Miranda Christ and Sam Gunn. Pseudorandom error-correcting codes. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024*, pages 325–347, Cham, 2024. Springer Nature Switzerland. i, ii, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 18, 19, 20, 28, 29
- [CGZ24] Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models. In *The Thirty Seventh Annual Conference on Learning Theory*, pages 1125–1139. PMLR, 2024. 1, 12
- [Che52] Herman Chernoff. A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the sum of Observations. *The Annals of Mathematical Statistics*, 23(4):493 – 507, 1952. 6
- [CM97] Christian Cachin and Ueli Maurer. Unconditional security against memory-bounded adversaries. In Burton S. Kaliski, editor, *Advances in Cryptology – CRYPTO ’97*, pages 292–306, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg. 5
- [Din01] Yan Zong Ding. Oblivious transfer in the bounded storage model. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, pages 155–170, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. 5
- [DQW23] Yevgeniy Dodis, Willy Quach, and Daniel Wichs. Speak much, remember little: Cryptography in the bounded storage model, revisited. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023*, pages 86–116, Cham, 2023. Springer Nature Switzerland. 5
- [GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, aug 1986. 12
- [GKLR21] Sumegha Garg, Pravesh K. Kothari, Pengda Liu, and Ran Raz. Memory-sample lower bounds for learning parity with noise, 2021. 4, 5, 23, 32
- [GLSS12] Dmitry Gavinsky, Shachar Lovett, Michael E. Saks, and Srikanth Srinivasan. A tail bound for read- k families of functions. *Random Structures & Algorithms*, 47, 2012. 24

- [GM24] Noah Golowich and Ankur Moitra. Edit distance robust watermarks for language models, 2024. [1](#), [2](#), [5](#), [13](#)
- [Gol00] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, USA, 2000. [27](#)
- [Gol11] Oded Goldreich. *Candidate One-Way Functions Based on Expander Graphs*, pages 76–87. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. [3](#), [5](#)
- [GRT18] Sumegha Garg, Ran Raz, and Avishay Tal. Extractor-based time-space lower bounds for learning. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018*, page 990–1002, New York, NY, USA, 2018. Association for Computing Machinery. [32](#)
- [Hoe94] Wassily Hoeffding. *Probability Inequalities for sums of Bounded Random Variables*, pages 409–426. Springer New York, New York, NY, 1994. [6](#)
- [KGW⁺23] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *International Conference on Machine Learning*, pages 17061–17084. PMLR, 2023. [12](#)
- [KRT17] Gillat Kol, Ran Raz, and Avishay Tal. Time-space hardness of learning sparse parities. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017*, page 1067–1080, New York, NY, USA, 2017. Association for Computing Machinery. [4](#), [5](#), [23](#), [32](#), [33](#)
- [LV17] Alex Lombardi and Vinod Vaikuntanathan. Minimizing the complexity of goldreich’s pseudorandom generator. Cryptology ePrint Archive, Paper 2017/277, 2017. <https://eprint.iacr.org/2017/277>. [15](#)
- [Mau92] Ueli M. Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5(1):53–66, 1992. [5](#)
- [OST22] Igor C. Oliveira, Rahul Santhanam, and Roei Tell. Expander-based cryptography meets natural proofs. *computational complexity*, 31(1):4, 2022. [5](#)
- [Raz18] Ran Raz. Fast learning requires good memory: A time-space lower bound for parity learning. *J. ACM*, 66(1), dec 2018. [4](#), [5](#), [11](#), [23](#), [32](#)
- [VV83] Umesh V. Vazirani and Vijay V. Vazirani. Trapdoor pseudo-random number generators, with applications to protocol design. In *24th Annual Symposium on Foundations of Computer Science (sfcs 1983)*, pages 23–30, 1983. [5](#)

A Time-space hardness of $\Theta(\log n)$ -sparse LPN

For our results in [Section 6](#), we need to confirm that any algorithm which solves the learning parity with noise where the secret has weight $\Theta(\log n)$ with non-negligible probability requires a significant amount of memory. We therefore begin by examining which parameter regimes we may

expect negligible success probability ([Raz18] [KRT17] [GKLR21] are generally not concerned with the distinction between negligible and non-negligible success probability).

Definition A.1 ([KRT17], this version appears in [GRT18]). *A set $T \subseteq \{0, 1\}^n$ is (ε, δ) -biased if there are at most $\delta \cdot 2^n$ elements $a \in \{0, 1\}^n$ with $|\mathbb{E}_{x \in_R T}[(-1)^{a \cdot x}]| > \varepsilon$, (where $a \cdot x$ denotes the inner product of a and x , modulo 2).*

Definition A.2 ([GKLR21]). *Let X, A be two finite sets. A matrix $M : A \times X \rightarrow \{-1, 1\}$ is a (k, ℓ) - L_2 -extractor with error 2^{-r} if for every nonnegative $f : X \rightarrow \mathbb{R}$ with $\|f\|_2/\|f\|_1 \leq 2^\ell$, there are at most $2^{-k} \cdot |A|$ rows $a \in A$ with*

$$\frac{|\langle M_a, f \rangle|}{\|f\|_1} \geq 2^{-r}$$

. Where $\|f\|_p = (\mathbb{E}_{x \leftarrow X}[|f(x)|^p])^{1/p}$, $M_a : X \rightarrow \mathbb{R}$ is the function corresponding to the a th row of M , and $\langle M_a, f \rangle = \mathbb{E}_{x \leftarrow X}[f(x) \cdot g(x)]$.

Lemma A.3 ([KRT17], this version appears in [GRT18]). *There exists a sufficiently small constant c such that the following holds. Let $\mathcal{S}_{\ell, n} = \{x \in \{0, 1\}^n : |x| = \ell\}$. If $\ell \leq n^{0.9}$, then $\mathcal{S}_{\ell, n}$ is an (ε, δ) -biased set for $\varepsilon = \ell^{-c\ell}$, $\delta = 2^{-cn/\ell^{0.01}}$.*

Lemma A.4 ([GRT18]). *Let $T \subseteq \{0, 1\}^n$ be an (ε, δ) -biased set, with $\varepsilon \geq \delta$. Then the matrix $M : T \times \{0, 1\}^n \rightarrow \{-1, 1\}$, defined by $M(a, x) = (-1)^{a \cdot x}$ is a (k, ℓ) - L_2 -extractor with error 2^{-r} , for $\ell = \Omega(\log(1/\delta))$, and $k = r = \Omega(\log(1/\varepsilon))$.*

Lemma A.5 (Theorem 5 of [GKLR21]). *Let $1/100 \leq c \leq \ln(2)/3$. Fix γ to be such that $(3c)/\ln(2) \leq \gamma^2 \leq 1$. Let X, A be two finite sets. Let $n = \log_2 |X|$. Let $M : A \times X \rightarrow \{-1, 1\}$ be a matrix which is a (k', ℓ') - L_2 -extractor with error 2^{-r} , for sufficiently large k', ℓ' , and r' , where $\ell' \leq n$. Let*

$$r := \min \left\{ \frac{r'}{2}, \frac{(1-\gamma)k'}{2}, \frac{(1-\gamma)\ell'}{2} - 1 \right\}.$$

Let B be a branching program, of length at most 2^r and width at most $2^{c \cdot k' \cdot \ell' / \varepsilon}$ for the learning problem that corresponds to the matrix M with error probability ε . Then the success probability of B is at most $O(2^{-r})$.

Lemma A.6. *No probabilistic polynomial time, $O(n \log^{0.99}(n)/\varepsilon)$ space algorithm solves the learning sparse parties problem with density $\Theta(\log n)$ and error rate $1/2 - \varepsilon$ with non-negligible probability.*

Proof. We begin by observing that any probabilistic polynomial time, $O(n \log^{0.99}(n)/\varepsilon)$ space algorithm implies a deterministic polynomial time, $O(n \log^{0.99}(n)/\varepsilon)$ space algorithm since a deterministic algorithm can simply use the first bit of a freshly sampled LPN sample any time it needs a random bit. Therefore, we only need to show that no deterministic polynomial time, $O(n \log^{0.99}(n)/\varepsilon)$ space algorithm solves the learning sparse parties with density $\Theta(\log n)$ and error rate $1/2 - \varepsilon$ with non-negligible probability.

Let $\ell = \Theta(\log n)$ be the weight of our LPN secret. **Lemma A.3** tells us that there exists a constant c such that $\mathcal{S}_{\ell, n}$ is an (ε, δ) -biased set where $\varepsilon = \ell^{-c\ell}$, $\delta = 2^{-cn/\ell^{0.01}}$. Since $\varepsilon \geq \delta$ (for sufficiently large n), by **Lemma A.4**, we see that the matrix M defined by $\mathcal{S}_{\ell, n}$ is a (k', ℓ') - L_2 -extractor with error $2^{-r'}$ for $\ell' = \Omega(n/\ell^{0.01})$, $k' = r' = \Omega(\ell \log(\ell))$. Notice that r', k', ℓ' are all $\Omega(\ell \log(\ell))$. Therefore,

by [Lemma A.5](#), we see that branching program which uses $o(k'\ell'/\varepsilon) = o(\ell \log(\ell)n/(\ell^{0.01}\varepsilon)) = O(n\ell^{0.99} \log(\ell)/\varepsilon)$ memory and $O(2^{\ell \log(\ell)})$ time has as a $O(2^{-\ell \log(\ell)}) = \text{negl}(n)$ of solving the learning parity with noise problem. \square

Finally, as pointed out in [\[KRT17\]](#), this gives us cryptography against a bounded space adversary. Since the inner product is a strong extractor, if we select our secret s from $\mathcal{S}_{\ell,n}$ and output $(a_1, a_1 \cdot s + e_1), \dots, (a_t, a_t \cdot s + e_t), a_{t+1}$ (where $t = \text{poly}(n)$, each a_i is sampled uniformly from $\{0, 1\}^n$, and each e_i is sampled from $\text{Ber}(1/2 - \varepsilon)$), no polynomial time, $o(n\ell^{0.99} \log(\ell)/\varepsilon)$ space algorithm can predict $a_{t+1} \cdot s$ with noticeable probability.

B Hypergeometric distribution lemma

We restate and prove [Lemma 2.5](#) and [Corollary 2.6](#).

Lemma 2.5. *If $0 \leq t \leq m \leq n$, $X \sim \text{Hyp}(n, m, t)$, then*

$$\frac{1}{2} + \frac{1}{2} \min_{\frac{m-t}{n} \leq p_i \leq \frac{m}{n-t}} \prod_{i=1}^t (1 - 2p_i) \leq \Pr[X \text{ is even}] \leq \frac{1}{2} + \frac{1}{2} \max_{\frac{m-t}{n} \leq p_i \leq \frac{m}{n-t}} \prod_{i=1}^t (1 - 2p_i).$$

Proof. Note that X is the number of special elements chosen if have n elements, of which m are special, and we choose t . Consider selecting the elements one by one. Let X_i be the indicator random variable denoting if the i th element chosen is special, let a_i denote the probability that we have an even number of special elements after i items are chosen. Let p_i denote $\Pr_{X_1, \dots, X_n} [X_1 \oplus \dots \oplus X_{i+1} = 1 | X_1 \oplus \dots \oplus X_i = 0]$. We will show by induction that $a_i = 1/2 + (1 + \prod_{j=1}^i (1 - 2p_j))$. When $i = 0$, this holds trivially. We now show the inductive case.

$$\begin{aligned} a_{i+1} &= \Pr_{X_1, \dots, X_n} [X_1 \oplus \dots \oplus X_{i+1} = 0] \\ &= \Pr_{X_1, \dots, X_n} [X_1 \oplus \dots \oplus X_{i+1} = 0 \cap X_1 \oplus \dots \oplus X_i = 0] \\ &\quad + \Pr_{X_1, \dots, X_n} [X_1 \oplus \dots \oplus X_{i+1} = 0 \cap X_1 \oplus \dots \oplus X_i = 1] \\ &= \Pr_{X_1, \dots, X_n} [X_1 \oplus \dots \oplus X_{i+1} = 0 | X_1 \oplus \dots \oplus X_i = 0] \cdot a_i \\ &\quad + \Pr_{X_1, \dots, X_n} [X_1 \oplus \dots \oplus X_{i+1} = 0 | X_1 \oplus \dots \oplus X_i = 1] \cdot (1 - a_i) \\ &= (1 - p_{i+1}) \cdot a_i + p_{i+1} \cdot (1 - a_i) \\ &= p_{i+1} + (1 - 2p_{i+1})a_i \\ &= \frac{1}{2} \left(1 + \prod_{j=1}^{i+1} (1 - 2p_j) \right) \end{aligned}$$

This concludes the inductive case. Notice that $\frac{m-t}{n} \leq p_i \leq \frac{m}{n-t}$ since every time we choose an element, the proportional of special elements to total elements left is at least $(m-t)/n$ and at most $m/(n-t)$. Therefore,

$$\min_{\frac{m-t}{n} \leq p_i \leq \frac{m}{n-t}} \frac{1}{2} \left(1 + \prod_{i=1}^t (1 - 2p_i) \right) \leq \Pr[X \text{ is even}] \leq \max_{\frac{m-t}{n} \leq p_i \leq \frac{m}{n-t}} \frac{1}{2} \left(1 + \prod_{i=1}^t (1 - 2p_i) \right).$$

The result follows by algebraic manipulation. \square

Corollary 2.6. *If $0 \leq t \leq m \leq n$, $X \sim \text{Hyp}(n, m, t)$ and p is a value maximizing $|1 - 2p|$ subject to $(m - t)/n \leq p \leq m/(n - t)$, then*

$$\Pr[X \text{ is even}] \leq \frac{1}{2} + \frac{1}{2}|1 - 2p|^t, \quad \text{and} \quad \Pr[X \text{ is odd}] \leq \frac{1}{2} + \frac{1}{2}|1 - 2p|^t .$$

Proof. Observe that

$$\max_{\frac{m-t}{n} \leq p_i \leq \frac{m}{n-t}} \prod_{i=1}^t (1 - 2p_i) \leq \max_{\frac{m-t}{n} \leq p_i \leq \frac{m}{n-t}} \prod_{i=1}^t |1 - 2p_i| \leq |1 - 2p|^t .$$

By [Lemma 2.5](#),

$$\Pr[X \text{ is even}] = \frac{1}{2} + \frac{1}{2} \max_{\frac{m-t}{n} \leq p_i \leq \frac{m}{n-t}} \prod_{i=1}^t (1 - 2p_i) \leq \frac{1}{2} + \frac{1}{2}|1 - 2p|^t .$$

Similarly,

$$\begin{aligned} \Pr[X \text{ is even}] &\geq \frac{1}{2} + \frac{1}{2} \min_{\frac{m-t}{n} \leq p_i \leq \frac{m}{n-t}} \prod_{i=1}^t (1 - 2p_i) \\ &= \frac{1}{2} + \frac{1}{2} \max_{\frac{m-t}{n} \leq p_i \leq \frac{m}{n-t}} - \prod_{i=1}^t (1 - 2p_i) \\ &= \frac{1}{2} - \frac{1}{2} \max_{\frac{m-t}{n} \leq p_i \leq \frac{m}{n-t}} \prod_{i=1}^t (1 - 2p_i) \\ &\geq \frac{1}{2} - \frac{1}{2}|1 - 2p|^t . \end{aligned}$$

This implies that the probability that X is odd is at most $1/2 + (1/2)|1 - 2p|^t$. \square