Computational Sample Complexity*

Scott E. Decatur[†] Oded Goldreich[‡] Dana Ron[§]

June 23, 1997

Abstract

In a variety of PAC learning models, a tradeoff between time and information seems to exist: with unlimited time, a small amount of information suffices, but with time restrictions, more information sometimes seems to be required. In addition, it has long been known that there are concept classes that can be learned in the absence of computational restrictions, but (under standard cryptographic assumptions) cannot be learned in polynomial time *regardless of sample size*. Yet, these results do not answer the question of whether there are classes for which learning from a small set of examples is infeasible, but becomes feasible when the learner has access to (polynomially) more examples.

To address this question, we introduce a new measure of learning complexity called *computational sample complexity* which represents the number of examples sufficient for *polynomial time* learning with respect to a fixed distribution. We then show concept classes that (under similar cryptographic assumptions) possess arbitrary sized gaps between their standard (information-theoretic) sample complexity and their computational sample complexity. We also demonstrate such gaps for learning from membership queries and learning from noisy examples.

KEYWORDS: Computational Learning Theory, Information vs. Efficient Computation, Pseudo-Random Functions, Error Correcting Codes, Wire-Tap Channel.

*An extended abstract of this work has appeared in the proceedings of the *10th COLT*, 1997.

†DIMACS Center, Rutgers University, Piscataway, NJ 08855. Part of this work was done while the author was at the Laboratory for Computer Science, MIT, supported by a grant from the Reed Foundation.

‡Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, ISRAEL. E-mail: oded@wisdom.weizmann.ac.il. On sabbatical leave at LCS, MIT.

§Laboratory for Computer Science, MIT, 545 Technology Sq., Cambridge, MA 02139. E-mail: danar@theory.lcs.mit.edu. Supported by an NSF postdoctoral fellowship.

1 Introduction

Perspective and Motivation. In this work, we examine the effects of computational restrictions on the number of examples needed for learning from random examples or membership queries. It has long been known that there are concept classes, containing only concepts which are implementable by "small" Boolean circuits, which can be learned in the absence of computational restrictions, yet cannot be learned (using any hypothesis class) in polynomial time (under standard cryptographic assumptions) [Val84, KV94, AK91, Kha93]. Yet, these results do not answer the question of whether there are classes for which learning from a small set of examples is infeasible, but becomes feasible when the learner has access to (polynomially) more examples. Such a phenomenon seems to be present in various learning problems (described below) and we focus on this tradeoff between information and computation.

The most common method of learning from examples in the PAC setting is through the use of Occam algorithms [BEHW87, BEHW89]. These are algorithms which take as input a set of labeled examples and output a concept from the target class which is consistent with the given set of examples. Blumer *et.al.* give an upper bound on the number of examples sufficient for an Occam algorithm to provide a good hypothesis. This bound depends on the PAC accuracy and confidence parameters and the Vapnik-Chervonenkis dimension (VC-Dimension) [VC71] of the target class. The general *lower* bound on the number of examples required for learning [EHKV89] nearly matches (within a logarithmic factor) the upper bound for Occam algorithms. Thus, the sample complexity for learning is essentially tight when we have an algorithm which finds a consistent concept from the target class.

While Occam algorithms exist for all classes[1], not all such algorithms are computationally efficient. Yet, for some of these classes learning is still feasible, although the *known* computationally efficient algorithms use more examples than does the Occam algorithm for the class. In these situations, computational restrictions appear to *impair* learning by requiring more data, but do not completely *preclude* learning. For example, it is NP-hard to find a k-term-DNF formula[2] consistent with a set of data labeled by a k-term-DNF formula [PV88]. The computationally efficient algorithm most commonly used for learning k-term-DNF works by finding a consistent hypothesis from an hypothesis class (kCNF) which strictly contains the target class. In using this larger class, the Occam algorithm requires a sample size dependent on n^k (the VC-Dimension of kCNF) as opposed to $k \cdot n$ (the VC-Dimension of k-term-DNF) as would be possible if the hypothesis class were k-term-DNF itself. Thus, although k-term-DNF learning is feasible, there is a gap between the sample size sufficient for learning k-term-DNF in the absence of computational restrictions and the sample size of known algorithms for computationally efficient learning.[3]

When the learner is allowed to make queries, we again see the phenomenon in which efficient learning seems to require more information than learning without such restrictions. One such example is the learning of deterministic finite automata (DFAs). Angluin's algorithm for this class [Ang87] can be viewed as drawing the standard Occam-sized sample and outputting a DFA consistent with it. But in order to efficiently find this consistent DFA, the algorithm makes many additional membership queries.

We also find computational restrictions to effect sample size when learning from examples

[1] In this work, we restrict our attention to classes of functions that can be represented by polynomially sized circuits.

[2] A k-term-DNF formula is a disjunctive normal form Boolean formula with at most constant k terms.

[3] Note that the possibility remains of there being an algorithm which learns using the "Occam number of examples" but does not learn by outputting a consistent k-term-DNF formula.

corrupted with noise. In the absence of computational restrictions, any PAC-learnable class can also be learned in the presence of classification noise rate $\eta = 1/2 - \gamma < 1/2$ using a factor of $\Theta(1/\gamma^2)$ more examples than the noise free case [Lai88, Tal94]. This increase is information theoretically required [Sim93, AD96]. Yet for many classes, when computation is restricted in the presence of noisy data, the sample complexity of *known* algorithms is increased by more than $\Theta(1/\gamma^2)$. Furthermore, this larger increase occurs even for classes which have computationally efficient noise-free Occam algorithms with optimal sample complexity, i.e., classes with no gap in their *noise-free* sample complexities. One very simple class exhibiting these properties is the class of monotone Boolean conjunctions.

Thus, it appears that in a variety of learning models (PAC, PAC with queries, and PAC with noise) there may exist a tradeoff between time and information – with unlimited time, a small amount of information will suffice, but with time restrictions, more information is required. None of the examples described above provably require the additional examples, yet researchers have been unable to close these gaps. In this work, we describe classes of functions for which we prove (based on cryptographic assumptions) a quantitative gap between the size of the sample required for learning a class and the size of the sample required for learning it efficiently.

SUMMARY OF OUR RESULTS. We focus our attention on *learning under the uniform distribution.* As discussed later in this section, this seems to be an appropriate and natural choice for demonstrating sample complexity gaps. Let $\mathcal{C} = \bigcup_n C_n$ be a concept class, where each C_n consists of Boolean functions over $\{0,1\}^n$. The (*Information Theoretic*) **Sample Complexity** of $\mathcal{C}$, denoted $\textsc{itsc}(\mathcal{C}; n, \epsilon)$, is the sample size (as a function of n and ϵ) needed for learning the class (without computational limitations) under the uniform distribution with approximation parameter ϵ, and confidence 9/10. The **Computational Sample Complexity** of $\mathcal{C}$, denoted $\textsc{csc}(\mathcal{C}; n, \epsilon)$, is the sample size needed for learning the class *in polynomial time* under the uniform distribution with approximation parameter ϵ, and confidence 9/10. In both cases, when the class is clear, we may omit it from the notation.

Definition 1.1 (admissible gap functions): *A function $g : \mathcal{N} \times \mathcal{R} \mapsto \mathcal{R}$ is called* admissible *if*

1. *$g(\cdot,\cdot)$ is polynomial-time computable.*
2. (bounded growth in n): *For every $\epsilon > 0$, $1 \le g(n,\epsilon) \le \mathrm{poly}(n)$.*
3. (monotonicity in ϵ): *For every $\epsilon, \epsilon' > 0$ such that $\epsilon < \epsilon'$, $g(n,\epsilon) \ge g(n,\epsilon')$;*
4. (smoothness in ϵ): *there exists a constant $a \ge 1$ so that for every $\epsilon > 0$, $g(n,\epsilon) \le a \cdot g(n, 2\epsilon)$.*

For example, the function $g(n,\epsilon) \stackrel{\mathrm{def}}{=} \frac{n^d}{\epsilon^{d'}}$ is admissible, for every $d, d' \ge 0$. Our main result is that any admissible function can serve as a gap between the sample complexity of some concept class and the computational sample complexity of the same class. That is:

Theorem 1.1 (basic model): *Let $g : \mathcal{N} \times \mathcal{R} \mapsto \mathcal{R}$ be an admissible function, and $k : \mathcal{N} \times \mathcal{R} \mapsto \mathcal{R}$ be of the form $k(n,\epsilon) = \frac{n^d}{\epsilon}$, where $d \ge 2$. Suppose that one-way functions exist.*[4] *Then there exists a concept class $\mathcal{C}$ which has sample complexity*

$$\textsc{itsc}(n,\epsilon) = \Theta(k(n,\epsilon))$$

and computational sample complexity

$$\textsc{csc}(n,\epsilon) = \Theta(g(n,\epsilon) \cdot k(n,\epsilon)) \, .$$

Furthermore, $\log_2 |C_n| = O(n^{d+1})$ and each function in C_n has a $\mathrm{poly}(n)$-size circuit.

[4] Here and in all our other conditional results, the computational sample complexity lower bounds hold for $\epsilon = 1/\mathrm{poly}(n)$ under standard complexity assumptions (i.e., the existence of one-way functions.) For smaller values of ϵ these bounds hold assuming slightly non-standard yet reasonable complexity assumptions.

	NOISE FREE		NOISE RATE $\eta = 0.25$	
	ITSC	CSC	ITSC	CSC
Item 1	$k(n,\epsilon)$	$g_1 \cdot g_2 \cdot k(n,\epsilon)$	$k(n,\epsilon)$	$g_1 \cdot g_2^2 \cdot k(n,\epsilon)$
Item 2	$k(n,\epsilon)$	$k(n,\epsilon)$	$k(n,\epsilon)$	$\frac{k(n,\epsilon)^2}{n^2}$

Figure 1: The results of Theorem 1.3 (with Θ-notation omitted). For general noise rate $\eta = 0.5 - \gamma \geq 0.25$, both ITSC and CSC increase linearly in $1/\gamma^2$.

In the above, and all subsequent theorems, one-way functions are merely used to construct pseudorandom functions [HILL, GGM86]. Assuming either that RSA is a one-way function or that the Diffie-Hellman Key Exchange is secure, one can construct pseudorandom functions in $\mathcal{NC}$ (cf., [NR95]), and so all of our "gap theorems" will follow with concept classes having $\mathcal{NC}$ circuits.

We next consider classification noise at rate $\eta < \frac{1}{2}$. That is, the label of each example is flipped with probability η, independently of all other examples. In this case we add $\gamma \stackrel{\rm def}{=} \frac{1}{2} - \eta > 0$ as a parameter to the sample complexity functions (e.g., $\text{ITSC}(\mathcal{C}; n, \epsilon, \gamma)$). We obtain:

Theorem 1.2 (noisy model): *Let $g : \mathcal{N} \times \mathcal{R} \mapsto \mathcal{R}$ be an admissible function, and $k : \mathcal{N} \times \mathcal{R}^2 \mapsto \mathcal{R}$ be of the form $k(n, \epsilon, \gamma) = \frac{n^2}{\epsilon\gamma^2}$. where $c \geq 2$. Suppose that one-way functions exist. Then there exists a concept class $\mathcal{C}$ which, in the presence of noise at rate $\eta = \frac{1}{2} - \gamma$, has sample complexity*

$$\text{ITSC}(n, \epsilon, \gamma) = \Theta(k(n, \epsilon, \gamma))$$

and computational sample complexity

$$\text{CSC}(n, \epsilon, \gamma) = \Theta(g(n, \epsilon) \cdot k(n, \epsilon, \gamma)) \ .$$

Furthermore, each function in C_n has a $\text{poly}(n)$*-size circuit.*

We stress that the above holds for every noise rate and in particular to the noise-free case (where $\eta = 0$ and $\gamma = 1/2$). Thus, we have for every $\gamma > 0$

$$\frac{\text{CSC}(n, \epsilon, \gamma)}{\text{ITSC}(n, \epsilon, \gamma)} = \Theta\left(\frac{\text{CSC}(n, \epsilon)}{\text{ITSC}(n, \epsilon)}\right) = \Theta\left(g(n, \epsilon)\right)$$

In particular, the computational sample complexity for moderate noise is of the same order of magnitude as in the noise-free case (i.e., $\text{CSC}(n, \epsilon, \frac{1}{4}) = \Theta(\text{CSC}(n, \epsilon))$). This stands in contrast to the following theorem in which the ratio between the two (i.e., $\frac{\text{CSC}(n,\epsilon,\frac{1}{4})}{\text{CSC}(n,\epsilon)}$) may be arbitrarily large, while $\text{ITSC}(n, \epsilon, \frac{1}{4}) = \Theta(\text{ITSC}(n, \epsilon))$ still holds (as it always does). See Figure 1.

Theorem 1.3 (noise, revisited): *Let $g_1, g_2 : \mathcal{N} \times \mathcal{R} \mapsto \mathcal{R}$ be admissible functions, and $k : \mathcal{N} \times \mathcal{R} \mapsto \mathcal{R}$ be of the form $k(n, \epsilon) = \frac{n^d}{\epsilon}$, where $d \geq 2$. Suppose that one-way functions exist. Then*

1. *there exists a concept class $\mathcal{C}$ which,*
   - *in the presence of noise at rate $\eta = \frac{1}{2} - \gamma \geq \frac{1}{4}$, has sample complexity*
     $$\text{ITSC}(n, \epsilon, \gamma) = \Theta(k(n, \epsilon)/\gamma^2)$$
     and computational sample complexity
     $$\text{CSC}(n, \epsilon, \gamma) = \Theta(g_1(n, \epsilon) \cdot (g_2(n, \epsilon))^2 \cdot k(n, \epsilon)/\gamma^2) \ ;$$
   - *whereas the noise-free complexities are*
     $$\text{ITSC}(n, \epsilon) = \Theta(k(n, \epsilon) \quad \textit{and} \quad \text{CSC}(n, \epsilon) = \Theta(g_1(n, \epsilon) \cdot g_2(n, \epsilon) \cdot k(n, \epsilon))$$
     respectively.

INFORMATION THEORETIC	COMPUTATIONAL MEASURES	
ITSC = ITQC	CQC	CSC
$k(n,\epsilon)$	$g_1 \cdot k(n,\epsilon)$	$g_1 \cdot g_2 \cdot k(n,\epsilon)$

Figure 2: The results of Theorem 1.4 (with Θ-notation omitted).

2. there exists a concept class $\mathcal{C}$ which,

- *in the presence of noise at rate $\eta = \frac{1}{2} - \gamma \geq \frac{1}{4}$, has sample complexity*
$$\mathrm{ITSC}(n,\epsilon,\gamma) = \Theta(k(n,\epsilon)/\gamma^2)$$
and computational sample complexity
$$\mathrm{CSC}(n,\epsilon,\gamma) = \Theta(\frac{k(n,\epsilon)^2}{n^2 \cdot \gamma^2})\ ;$$
- *whereas the noise-free complexities are*
$$\mathrm{CSC}(n,\epsilon) = \Theta(\mathrm{ITSC}(n,\epsilon)) = \Theta(k(n,\epsilon))\ .$$

Furthermore, each function in C_n has a poly(n)*-size circuit.*

Theorem 1.2 follows as a special case of Item 1 by setting $g_2 \equiv 1$. Using Item 2 we get that for every $\alpha > 0$ and for every $\gamma \leq 1/4$, $\mathrm{CSC}(n,\epsilon,\gamma) = \Omega(\mathrm{CSC}(n,\epsilon)^{2-\alpha}/\gamma^2)$.

We now turn to learning with membership queries. The (*Information Theoretic*) Query Complexity of $\mathcal{C}$, denoted $\mathrm{ITQC}(\mathcal{C};n,\epsilon)$, is the number of membership queries (as a function of n and ϵ) needed for learning the class (without computational limitations) under the uniform distribution with approximation parameter ϵ, and confidence 9/10. The Computational Query Complexity of $\mathcal{C}$, denoted $\mathrm{CQC}(\mathcal{C};n,\epsilon)$, is the number of queries needed for learning the class *in polynomial time* under the uniform distribution with approximation parameter ϵ, and confidence 9/10. We obtain (see also Figure 2):

Theorem 1.4 (query model): *Let $g_1, g_2 : \mathcal{N} \times \mathcal{R} \mapsto \mathcal{R}$ be two admissible functions, and $k : \mathcal{N} \times \mathcal{R} \mapsto \mathcal{R}$ be of the form $k(n,\epsilon) = \frac{n^d}{\epsilon}$, where $d \geq 2$. Suppose that one-way functions exist. Then there exists a concept class $\mathcal{C}$ which has query complexity*
$$\mathrm{ITQC}(n,\epsilon) = \Theta(k(n,\epsilon)) = \Theta(\mathrm{ITSC}(n,\epsilon))$$
computational query complexity
$$\mathrm{CQC}(n,\epsilon) = \Theta(g_1(n,\epsilon) \cdot k(n,\epsilon))$$
and computational sample complexity
$$\mathrm{CSC}(n,\epsilon) = \Theta(g_1(n,\epsilon) \cdot g_2(n,\epsilon) \cdot k(n,\epsilon))\ .$$
Furthermore, each function in C_n has a poly(n)*-size circuit.*

Note that we may set $g_2 \equiv 1$ and obtain
$$\mathrm{CSC}(n,\epsilon) = \Theta(\mathrm{CQC}(n,\epsilon)) = \Theta(g_1(n,\epsilon) \cdot \mathrm{ITQC}(n,\epsilon))$$
(and $\mathrm{ITSC}(n,\epsilon) = \Theta(\mathrm{ITQC}(n,\epsilon))$).

UNIFORM VS. DISTRIBUTION-FREE LEARNING. Above, we show that in a variety of settings there exists a concept class exhibiting a sample complexity gap when learning occurs with respect to the uniform distribution. We note that in all our theorems the information theoretic upper bounds hold, within a factor of n, with respect to distribution-free learning.[5] Thus, there exist concept

[5] In Theorem 1.4 the upper bounds hold in the distribution-free case, without any extra factor.

classes for which efficient learning *under the uniform distribution* (is possible but) requires vastly larger sample sizes than the *distribution-free* information theoretic upper bound.

One may wonder whether there exists a concept class exhibiting similar sample complexity gaps with respect to *every* distribution. Clearly, degenerate distributions preclude such results. Alternatively, one may wonder whether, for every distribution, there exists a class that exhibits sample complexity gaps. Again, such results are precluded by degenerate distributions. Thus, we believe that an appropriate goal is to demonstrate gaps on fixed distributions, the uniform distribution being the most natural and well studied.[6]

Although the above notion of a gap cannot exist for a single concept class across all distributions, a different notion of distribution-free gap can exist. Specifically, we may consider a gap between: (1) an upper bound on the information-theoretic distribution-free sample complexity; and (2) a lower bound on the distribution free sample complexity of an efficient learner that is *tight* (i.e., has a matching upper bound). More precisely, let the (*Distribution-Free Information Theoretic*) **Sample Complexity** of $\mathcal{C}$, denoted $\mathcal{ITSC}(\mathcal{C}; n, \epsilon)$, be the sample size (as a function of n and ϵ) needed for learning the class (without computational limitations); and let the *Distribution-Free* **Computational Sample Complexity** of $\mathcal{C}$, denoted $\mathcal{CSC}(\mathcal{C}; n, \epsilon)$, be the sample size needed for learning the class *in polynomial time*. We stress that an upper bound for any of these measures refers to all possible distributions, whereas a lower bound merely refers to one (possibly "pathological") distribution. In fact, such pathological distributions are used in the result below.

Theorem 1.5 (distribution-free): *Let p be a polynomial so that $p(n) \geq n$, and suppose that one-way functions exist. Then there exists a concept class $\mathcal{C}$ so that $\mathcal{ITSC}(n, \epsilon) = O(n/\epsilon)$ whereas $\mathcal{CSC}(n, \epsilon) = \Theta(p(n)/\epsilon)$. Furthermore, each function in C_n has a* poly(n)*-size circuit.*

Note that the gap shown in the theorem is polynomially in n (independent of ϵ). Thus, we do not get arbitrary admissible gaps as in Theorem 1.1. We note that the computational sample complexity *under the uniform distribution* for this class is $\Theta(p(n) \cdot \min\{\log(1/\epsilon), \log p(n)\})$.

TECHNIQUES. The basic idea is to consider concepts which consist of two parts: The first part of the concept is determined by a pseudorandom function (cf., [GGM86]), while the second part encodes the seed of such a function. Since it is infeasible to infer a pseudorandom function, the computational-bounded learner is forced to retrieve the seed of the function which is sparsely encoded in the second part. This sparse encoding makes retrieval very costly in terms of sample complexity; yet, the computationally-UNbounded learner is not effected by it.

The basic idea described above suffices for establishing a gap between the computational sample complexity and the information-theoretic sample complexity for a fixed ϵ. Additional ideas are required in order to have a construction which works for any ϵ, and for which one may provide tight (up to a constant factor) bounds on each of the two complexities. One of these ideas is the construction of concept classes, called *equalizers*, for which the computational sample complexity upper bound is of the same order as the information-theoretic lower bound. An result of this form follows:

Theorem 1.6 (equalizers): *Let $p(\cdot)$ be any polynomial.*

1. (noisy-sample equalizer): *There exists a concept class $\mathcal{S} = \cup_n S_n$, with concepts realizable by polynomial-size circuits, such that*
$$\text{ITSC}(\mathcal{S}; n, \epsilon, \gamma) = \Theta(\text{CSC}(\mathcal{S}; n, \epsilon, \gamma)) = \Theta(p(n)/\epsilon\gamma^2) .$$

[6] We note that our techniques may be used to show similar sample complexity gaps on distributions other than the uniform distribution.

2. (query equalizer): *There exists a concept class $\mathcal{S} = \cup_n S_n$, with concepts realizable by polynomial-size circuits, such that*

$$\textsc{itqc}(\mathcal{S}; n, \epsilon) = \Theta(\textsc{csc}(\mathcal{S}; n, \epsilon)) = \Theta(p(n)/\epsilon) .$$

Another idea used in our proofs is the introduction and utilization of a novel (probabilistic) coding scheme. In addition to the standard coding theoretic requirements, this scheme has the property that any constant fraction of the bits in the (randomized) codeword yields no information about the message being encoded. We also use this coding scheme to obtain efficient constructions for the *Wire-Tap Channel* Problem (cf., [Wyn75]) – see Proposition 2.2. We believe that this probabilistic coding scheme is of independent interest.

ORGANIZATION. After introducing the cryptographic tools we shall nee, we establish the separation of computational sample complexity from (IT) sample complexity in the basic model. This result (i.e., Theorem 1.1) may be derived as a special case of the other results, but we chose to present a self-contained and simpler proof of the separation in the basic model: All that is needed is our new coding scheme (presented in Section 2), and the basic construction (presented in Section 3).

To establish separation in the noise and query models, we use a more general construction. This construction utilizes the Great Equalizer (of Theorem 1.6 presented in Section 4). The general construction itself is presented in Section 5 and is used to derive Theorems 1.1 through 1.4.

Theorem 1.5 is proven in Section 6. We note that this proof is much simpler than any other proof in the paper and that it can be read without reading any of the other sections.

2 Cryptographic Tools

In subsection 2.1 we review known definitions and results regarding pseudorandom functions. In subsection 2.2 we present a computationally efficient (randomized) coding scheme which on top of the standard error-correction features has a secrecy feature. Specifically, a small fraction of (the uncorrupted) bits of the codeword yield no information about the message being encoded.

2.1 Pseudorandom Functions

Loosely speaking, pseudorandom functions are easy to select and evaluate, yet look as random functions to any computationally restricted observer who may obtain their value at inputs of its choice.

Definition 2.1 (pseudorandom functions [GGM86]): *Let $\ell : \mathcal{N} \mapsto \mathcal{N}$ be a polynomially-bounded length function, and $\mathcal{F} = \{F_n : n \in \mathcal{N}\}$ where $F_n = \{f_\alpha : \alpha \in \{0,1\}^n\}$ is a (multi)set of 2^n Boolean functions over the domain $\{0,1\}^{\ell(n)}$. The family $\mathcal{F}$ is called a* pseudorandom family *if*

- Easy to Evaluate: *There exist a polynomial-time algorithm A so that $A(\alpha, x) = f_\alpha(x)$, for every $\alpha \in \{0,1\}^*$ and $x \in \{0,1\}^{\ell(|\alpha|)}$. The string α is called the* seed *of f_α.*
- Pseudorandomness: *For every probabilistic polynomial-time oracle machine M, every positive polynomial p and all sufficiently large n's*

$$|\Pr_{f \in F_n}(M^f(1^n)=1) - \Pr_{g \in R_n}(M^g(1^n)=1)| < \frac{1}{p(n)}$$

where R_n denotes the set of all $(2^{2^{\ell(n)}})$ Boolean functions over the domain $\{0,1\}^{\ell(n)}$.

Pseudorandom functions exists if and only if there exist one-way functions (cf., [GGM86] and [HILL]). Pseudorandom functions which can be evaluated by $\mathcal{NC}$ circuits (one circuit per each function) exist[7], assuming either that RSA is a one-way function or that the Diffie-Hellman Key Exchange is secure (cf., [NR95]).

2.2 A Probabilistic Coding Scheme

We present an efficient probabilistic encoding scheme having constant *rate* (information/codeword ratio), constant (efficient) *error-correction* capability for which a (small) constant fraction of the codeword bits yield no information about the plain message. Note that a scheme as described above cannot be deterministic (as each bit in a deterministic coding scheme carries information).

Theorem 2.1 *There exist constants $c_{\text{rate}}, c_{\text{err}}, c_{\text{sec}} < 1$ and a pair of probabilistic polynomial-time algorithms, (E, D), so that*

1. Constant Rate*: $|E(x)| = |x|/c_{\text{rate}}$, for all $x \in \{0,1\}^*$.*
2. Linear Error Correction*: for every $x \in \{0,1\}^*$ and every $e \in \{0,1\}^{|E(x)|}$ which has at most $c_{\text{err}} \cdot |E(x)|$ ones,*
$$\Pr(D(E(x) \oplus e) = x) \ = \ 1$$
where $\alpha \oplus \beta$ denotes the bit-by-bit exclusive-or of the strings α and β. Algorithm D is deterministic.
3. Partial Secrecy*: Loosely speaking, a substring containing $c_{\text{sec}} \cdot |E(x)|$ bits of $E(x)$ does not yield information on x. Namely, let I be a subset of $\{1, ..., |\alpha|\}$, and let α_I denote the substring of α corresponding to the bits at locations $i \in I$. Then for every $n \in \mathcal{N}$, $m = n/c_{\text{rate}}$, $x, y \in \{0,1\}^n$, $I \in \{J \subset \{1, ..., m\} : |J| \leq c_{\text{sec}} \cdot m\}$, and $\alpha \in \{0,1\}^{|I|}$,*
$$\Pr(E(x)_I = \alpha) \ = \ \Pr(E(y)_I = \alpha)$$
Furthermore, $E(x)_I$ is uniformly distributed over $\{0,1\}^{|I|}$.

In addition, on input x, algorithm E uses $O(|x|)$ coin tosses.

Items 1 and 2 are standard requirements of Coding Theory, first met by Justesen [Jus72]. What is non-standard in the above is Item 3. Indeed, Item 3 is impossible if one insists that the encoding algorithm (i.e., E) be deterministic.

Proof: Using a "nice" error correcting code, the key idea is to encode the information by first augmenting it by a sufficiently long random padding. To demonstrate this idea, consider an $2n$-by-m matrix M defining a constant-rate/linear-error-correction (linear) code. That is, the string $z \in \{0,1\}^{2n}$ is encoded by $z \cdot M$. Further suppose that the submatrix defined by the last n rows of M and any $c_{\text{sec}} \cdot m$ of its columns is of full-rank (i.e., rank $c_{\text{sec}} \cdot m$). Then, we define the following probabilistic coding, E, of strings of length n. To encode $x \in \{0,1\}^n$, we first uniformly select $y \in \{0,1\}^n$, let $z = xy$ and output $E(x) = z \cdot M$. Clearly, the error-correction features of M are inherited by E. To see that the secrecy requirement holds consider any sequence of $c_{\text{sec}} \cdot m$ bits in $E(x)$. The contents of these bit locations is the product of z by the corresponding columns in M; that is, $z \cdot M' = x \cdot A + y \cdot B$, where M' denotes the submatrix corresponding to these columns in M, and A (resp., B) is the matrix resulting by taking the first (resp., last) n rows of M'. By

[7] Actually, these circuits can be constructed in polynomial-time given the seed of the function.

hypothesis B is full rank, and therefore $y \cdot B$ is uniformly distributed (and so is $z \cdot M'$ regardless of x).

What is missing in the above is a specific construction satisfying the hypothesis as well as allowing efficient decoding. Such a construction can be obtained by mimicking Justesen's construction [Jus72]. Recall that Justesen's Code is obtained by composing two codes: Specifically, an *outer* linear code over an n-symbol alphabet is composed with an *inner* random linear code.[8] The outer code is obtained by viewing the message as the coefficients of a polynomial of degree $t-1$ over a field with $\approx 3t$ elements, and letting the codeword consists of the values of this polynomial at all field elements. Using the Berlekamp-Welch Algorithm [BW86], one can efficiently retrieve the information from a codeword provided that at most t of the symbols (i.e., the values at field elements) were corrupted. We obtain a variation of this outer-code as follows: Given $x \in \{0,1\}^n$, we set $t \stackrel{\rm def}{=} 2n/\log_2(2n)$, and view x as a sequence of $\frac{t}{2}$ elements in GF$(3t)$.[9] We uniformly select $y \in \{0,1\}^n$ and view it as another sequence of $\frac{t}{2}$ elements in GF$(3t)$. We consider the degree $t-1$ polynomial defined by these t elements, where x corresponds to the high-order coefficients and y to the low order ones. Clearly, we preserve the error-correcting features of the original outer code. Furthermore, any $t/2$ symbols of the codeword yield no information about x. To see this, note that the values of these $t/2$ locations are obtained by multiplying a t-by-$t/2$ Vandermonde with the coefficients of the polynomial. We can rewrite the product as the sum of two products the first being the product of a $t/2$-by-$t/2$ Vandermonde with the low order coefficients. Thus, a uniform distribution on these coefficients (represented by y) yields a uniformly distributed result (regardless of x).

Next, we obtain an analogue of the inner code used in Justesen's construction. Here the aim is to encode information of length $\ell \stackrel{\rm def}{=} \log_2 3t$ (i.e., the representation of an element in GF$(3t)$) using codewords of length $O(\ell)$. Hence, we do not need an efficient decoding algorithm, since Maximum Likelihood Decoding via exhaustive search is affordable (as $2^\ell = O(t) = O(n)$). Furthermore, any code which can be specified by $\log(n)$ many bits will do (as we can try and check all possibilities in poly(n)-time), which means that we can use a randomized argument provided that it utilizes only $\log(n)$ random bits. For example, we may use a linear code specified by a (random) 2ℓ-by-4ℓ Toeplitz matrix.[10] Using a probabilistic argument one can show that with positive probability such a random matrix yields a "nice" code as required in the motivating discussion.[11] In the rest of the discussion, one such good Toeplitz matrix is fixed.

We now get to the final step in mimicking Justesen's construction: the composition of the two codes. Recall that we want to encode $x \in \{0,1\}^n$, and that using a random string $y \in \{0,1\}^n$ we have generated a sequence of $3t$ values in GF$(3t)$, denoted $x_1, ..., x_{3t}$, each represented by a binary string of length ℓ. (This was done by the outer code.) Now, using the inner code (i.e., the Toeplitz matrix) and additional $3t$ random ℓ-bit strings, denoted $y_1, ..., y_{3t}$, we encode each of the above x_i's by a 4ℓ-bit long string. Specifically, x_i is encoded by the product of the Toeplitz matrix with the vector $x_i y_i$.

Clearly, we preserve the error-correcting features of Justesen's construction [Jus72]. The Secrecy condition is shown analogously to the way in which the Error Correction feature is established in [Jus72]. Specifically, we consider the partition of the codeword into consecutive 4ℓ-bit long

[8] Our presentation of Justesen's Code is inaccurate but suffices for our purposes.

[9] Here we assume that $3t$ is a prime power. Otherwise, we use the first prime power greater than $3t$. Clearly, this has a negligible effect on the construction.

[10] A Toeplitz matrix, $T = (t_{i,j})$, satisfies $t_{i,j} = t_{i+1,j+1}$, for every i, j.

[11] The proof uses the fact that any (non-zero) linear combination of rows (columns) in a random Toeplitz matrix is uniformly distributed.

subsequences corresponding to the codewords of the inner code. Given a set I of locations (as in the secrecy requirement), we consider the relative locations in each subsequence, denoting the induced locations in the i^{th} subsequence by I_i. We classify the subsequences into two categories depending on whether the size of the induced I_i is above the secrecy threshold for the inner code or not. By a counting argument, only a small fraction of the subsequences have I_i's above the threshold. For the rest we use the Secrecy feature of the inner code to state that no information is revealed about the corresponding x_i's. Using the Secrecy feature of the outer code, we conclude that no information is revealed about x. ■

EFFICIENT CODING FOR THE WIRE-TAP CHANNEL PROBLEM: Using Theorem 2.1, we obtain an efficient coding scheme for (a strong version of) the *Wire-Tap Channel* Problem (cf., [Wyn75]). Actually, we consider a seemingly harder version introduced by Csiszár and Körner [CK78]. To the best of our knowledge no *computationally efficient* coding scheme was presented for this problem before.[12]

Proposition 2.2 *Let (E, D) be a coding scheme as in Theorem 2.1 and let $\mathrm{BSC}_p(\alpha)$ be a random process which represents the transmission of a string α over a Binary Symmetric Channel with crossover probability*[13] *p. Then,*

1. Error Correction*: For every $x \in \{0,1\}^*$*

$$\Pr(D(\mathrm{BSC}_{\frac{c_{\text{err}}}{2}}(E(x))) = x) \;=\; 1 - \exp(-\Omega(|x|))$$

2. Secrecy*: For every $x \in \{0,1\}^*$*

$$\sum_{\alpha \in \{0,1\}^{|E(x)|}} \left| \Pr(\mathrm{BSC}_{\frac{1}{2}-\frac{c_{\text{sec}}}{4}}(E(x)) = \alpha) \;-\; 2^{-|E(x)|} \right|$$

is exponentially vanishing in $|x|$.

Proof: Item 1 follows by observing that, with overwhelming high probability, the channel complements less than a $c_{\text{err}}/2$ fraction of the bits of the codeword. Item 2 follows by representing $\mathrm{BSC}_{(1-\gamma)/2}(\alpha)$ as a two-stage process: In the first stage each bit of α is *set* (to its current value) with probability γ, independently of the other bits. In the second stage each bit which was not set in the first stage, is assigned a uniformly chosen value in $\{0,1\}$. Next, we observe that, with overwhelming high probability, at most $2\gamma|E(x)| = c_{\text{sec}}|E(x)|$ bits were set in the first stage. Suppose we are in this case. Then, applying Item 3 of Theorem 2.1, the bits set in Stage 1 are uniformly distributed regardless of x, and due to Stage 2 the un-set bits are also random. ■

Remark 2.3 *The above proof can be easily adapted to assert that, with overwhelming high probability, no information about x is revealed when obtaining both $\frac{c_{\text{sec}}}{2} \cdot |E(x)|$ of the bits of $E(x)$ as well as the entire $\mathrm{BSC}_{\frac{1}{2}-\frac{c_{\text{sec}}}{8}}(E(x))$.*

[12] We note that Maurer has shown that this version of the problem can be reduced to the original one by using bi-directional communiaction [Mau91]. Crépeau (private comm., April 1997) has informed us that, using the techniques in [BBCM95, CM97], one may obtain an alternative efficient solution to the original Wire-Tap Channel Problem again by using bi-directional communiaction.

[13] The *crossover probability* is the probability that a bit is complemented in the transmission process.

3 Proof of Theorem 1.1

We start by describing a construction which satisfies the gap requirement of Theorem 1.1 for a fixed ϵ, say $\epsilon = 0.1$. That is, we only show that there exists a concept class $\mathcal{C}$, which for $\epsilon = 0.1$, has sample complexity $\text{ITSC}(n, \epsilon) = \Theta(k(n, \epsilon))$ and computational sample complexity $\text{CSC}(n, \epsilon) = \Theta(g(n, \epsilon) \cdot k(n, \epsilon))$. The construction is later generalized to handle variable ϵ.

3.1 Motivation: Construction for constant ϵ

We view a function $f \in C_n$ as an array of 2^n bits. This array is divided into the following three (consecutive) *slices* which have sizes 2^{n-1}, 2^{n-2} and 2^{n-2}, respectively.

Slice I: This slice, called the *pseudorandom slice*, is determined by a pseudorandom function $f_s : \{0,1\}^{n-1} \to \{0,1\}$, where the seed s is of length n. (See subsection 2.1.)

Slice II: This slice, called the *seed encoder*, is determined by the abovementioned seed s and an additional string r of length $O(n)$. More precisely, first we employ the probabilistic encoding scheme of subsection 2.2 to encode the message s using r as the randomness required by the scheme. The result is a codeword of length $m \stackrel{\text{def}}{=} O(n)$. Next we repeat each bit of the codeword in $\frac{2^{n-2}}{g(n,0.1) \cdot k(n,0.1)}$ specified locations. All other locations in this slice are set to zero.

Slice III: This slice, called the *sample equalizer*, is determined by a binary string u of length $k(n, 0.1)$. The slice consists entirely of $k(n, 0.1)$ blocks of equal length, each repeating the corresponding bit of u. The purpose of this slice is to dominate the (information theoretic) sample complexity, and allow us to easily derive tight bounds on it.

INFORMATION THEORETIC BOUNDS. Applying Occam's Razor [BEHW87] to the class $\mathcal{C}$, we obtain $\text{ITSC}(\mathcal{C}; n, 0.1) = O(\log |C_n|) = O(n + O(n) + k(n, 0.1)) = O(k(n, 0.1))$ where the last equality is due to $k(n, 0.1) > n$. On the other hand, in order to learn a function in the class with error at most 0.1, it is necessary to learn Slice III with error at most 0.4. Thus, by virtue of Slice III alone, we have

$$\text{ITSC}(\mathcal{C}; n, 0.1) \geq \text{ITSC}(\text{Slice III}; n, 0.4) \geq 0.2 \cdot k(n, 0.1)$$

where the last inequality is due to the fact that learning a random string with error ϵ requires obtaining at least $1 - 2\epsilon$ of its bits. Thus, we have established the desired information-theoretic bounds. We now turn to analyze the computational sample complexity.

COMPUTATIONAL LOWER BOUND. The computationally bounded learner cannot learn Slice I from examples (or even queries) in the slice. Still, it must learn Slice I with error at most 0.2. Hence, the role of Slice I is to force the computationally bounded learner to obtain the function's seed from Slice II. By Item 3 of Theorem 2.1, in order to attain any information (from Slice II) regarding the seed, the learner must obtain $\Omega(n)$ bits of the codeword (residing in Slice II). By Item 1 of Theorem 2.1, this means obtaining a constant fraction of the bits of the codeword. Recall that the probability of getting any bit in the codeword is $\frac{O(n)}{g(n,0.1) \cdot k(n,0.1)}$. Therefore, by a Chernoff Bound, for every fraction $\alpha < 1$, there exists a constant $\beta < 1$, such that the probability of obtaining a fraction α of the codeword given $(\beta \cdot g(n, 0.1) \cdot k(n, 0.1))$ examples, is exponentially small. Thus, $\text{CSC}(\mathcal{C}; n, 0.1) = \Omega(g(n, 0.1) \cdot k(n, 0.1))$.

COMPUTATIONAL UPPER BOUND. By Chernoff Bound a sample of $O(g(n, 0.1) \cdot k(n, 0.1))$ examples contains, with overwhelmingly high probability, an occurrence of each bit of the codeword of the seed. Thus, by (a special case of) Item 2 of Theorem 2.1, the learner can efficiently retrieve the seed and so derive all of Slices I and II of the concept. However, by $g(n, 0.1) \geq 1$, the above sample will also allow obtaining (with high probability) all but at most a 0.1 fraction of the bits in Slice III, and thus $\text{CSC}(\mathcal{C}; n, 0.1) = O(g(n, 0.1) \cdot k(n, 0.1))$.

3.2 General Construction – Variable ϵ

We adopt the basic structure of the construction above, except that each of the three slices is further subpartitioned into *blocks*. Specifically, each slice has $t \stackrel{\text{def}}{=} n - \log_2 O(n)$ (consecutive) *blocks*, so that each block corresponds to a different possible value of $\epsilon = 2^{-i}$, for $i = 1, ..., t$. We start with a detailed description of each of the three slices (see Figure 1).

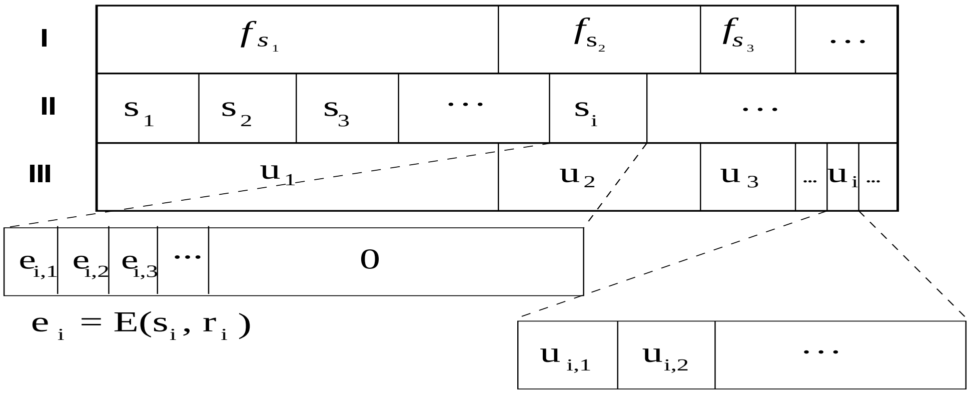

Figure 1: Construction for Theorem 1.1

SLICE I: THE PSEUDORANDOM PART. The i^{th} block has size 2^{n-1-i} and is determined by a pseudorandom function $f_{s_i} : \{0,1\}^{n-1-i} \to \{0,1\}$, where the seed s_i is of length n. Note that the blocks are shrinking in size, and that they are all pseudorandom.

SLICE II: THE SEEDS ENCODER. Here the blocks are of equal size $B \stackrel{\text{def}}{=} \frac{1}{t} \cdot 2^{n-2}$. The i^{th} block encodes the i^{th} seed, s_i, using an additional string r_i of length $O(n)$. Let e_i be the codeword obtained by employing the probabilistic encoding scheme (of Theorem 2.1) on input s_i using randomness r_i. Recall that $m \stackrel{\text{def}}{=} |e_i| = O(n)$. The i^{th} block is further divided into m (consecutive) *information fields*, each of size $\frac{2^{n-2}}{g(n,2^{-i}) \cdot k(n,2^{-i})}$, and an additional *empty field* (of size $B - \frac{m \cdot 2^{n-2}}{g(n,2^{-i}) \cdot k(n,2^{-i})}$). Thus the m information fields have relative density $\frac{m \cdot t}{g(n,2^{-i}) \cdot k(n,2^{-i})} = \Theta(\frac{n^2}{g(n,2^{-i}) \cdot k(n,2^{-i})})$ with respect to the total size B of the block. All bits in the j^{th} information field are set to the value of the j^{th} bit of e_i, and all bits in the empty field are set to zero.

SLICE III: THE SAMPLE EQUALIZER. The i^{th} block has size $B_i \stackrel{\text{def}}{=} 2^{n-2-i}$ and is determined by a binary string u_i of length $K \stackrel{\text{def}}{=} 2^{-i} \cdot k(n, 2^{-i}) = n^d$. The i^{th} block is further divided into K (consecutive) sub-blocks, each of size $\frac{B_i}{K}$. All bits in the j^{th} sub-block are set to the value of the j^{th} bit of u_i. Note that the blocks are shrinking in size, and that for each block it is possible to obtain most bits in the block by viewing $\Theta(K)$ random examples residing in it.

We first observe that Slice III above gives rise to a concept class for which tight bounds, of the form $k(n, \epsilon) = \text{poly}(n)/\epsilon$, on the information theoretic and computational sample complexities, can be given.

Proposition 3.1 *For* Slice III *described above we have:*

1. $\text{ITSC}(\text{Slice III}; n, 4\epsilon) = \Omega(k(n,\epsilon))$.
2. $\text{CSC}(\text{Slice III}; n, \epsilon) = O(k(n,\epsilon))$.

Proof: *Item 1.* Let $i \stackrel{\text{def}}{=} \lfloor \log_2(1/8\epsilon) \rfloor$. In order to learn Slice III with error at most 4ϵ, one must learn the i^{th} block reasonably well. Specifically, one must obtain examples from at least half of the K sub-blocks of the i^{th} block, and hence must have at least $K/2$ examples in the i^{th} block. By Chernoff Bound, this implies that the total number of random samples must be at least $2^i \cdot \frac{K}{4} = 2^{i-2}\epsilon \cdot k(n,\epsilon) > 2^{-7} \cdot k(n,\epsilon)$. Item 1 follows.

Item 2. On the other hand, we consider the fraction of the third slice that is determined (with constant probability close to 1) given a sample of $16 \cdot k(n,\epsilon) = 16K/\epsilon$ random examples. It suffices to show that the total area left undetermined in the first $\ell \stackrel{\text{def}}{=} \lceil \log_2(4/\epsilon) \rceil$ blocks is at most an $\epsilon/2$ fraction of the total domain (since the remaining blocks cover at most an $\epsilon/2$ fraction of the total). Fixing any $i \le \ell$, we consider the expected number of sub-blocks determined in the i^{th} blocks (out of the total K sub-blocks). A sub-block is determined if and only if we obtain a sample in it, and the probability for the latter event not to occur in $16K/\epsilon$ trials is

$$\left(1 - \frac{2^{-i}}{K}\right)^{16K/\epsilon} = \exp\left(-\frac{16}{2^i\epsilon}\right) \tag{1}$$

It follows that the expected fraction of bits which are not determined in the first ℓ blocks is bounded above by

$$\begin{aligned}
\sum_{i=1}^{\ell} 2^{-i} \cdot 2^{-\frac{16}{2^i\epsilon}} &= \sum_{j=0}^{\ell-1} 2^{-(\ell-j)} \cdot 2^{-\frac{16}{2^{\ell-j}\cdot 2^{-\ell+2}}} && (2)\\
&= 2^{-\ell} \cdot \sum_{j=0}^{\ell-1} 2^j \cdot 2^{-2^{j+2}} && (3)\\
&< \frac{\epsilon}{4} \cdot 2^{-4} \sum_{j=0}^{\infty} 2^{-j} && (4)\\
&< \frac{\epsilon}{32} && (5)
\end{aligned}$$

where Eq. (4) follows from the fact that $\forall j \ge 0$, $2^{j-4\cdot 2^j} \le 2^{-4} \cdot 2^{-j}$. Item 2 follows. ■

Lemma 3.1 *The concept class described above has* (information theoretic) *sample complexity* $\text{ITSC}(n,\epsilon) = \Theta(k(n,\epsilon))$.

Proof: Clearly, it suffices to learn each of the three slices with error ϵ. Applying Occam's Razor [BEHW87] to Slices I and II of the class, we obtain

$$\text{ITSC}(\text{Slices I and II}; n, \epsilon) = O(n^2/\epsilon) = O(k(n,\epsilon))$$

where the last equality is due to the hypothesis regarding the function $k(\cdot,\cdot)$ (i.e., that it is $\Omega(n^2/\epsilon)$). Using Item 2 of Proposition 3.1, we obtain $\text{ITSC}(\text{Slice III}; n, \epsilon) \le \text{CSC}(\text{Slice III}; n, \epsilon) = O(k(n,\epsilon))$, and $\text{ITSC}(n,\epsilon) = O(k(n,\epsilon))$ follows. (Note that in order to obtain the desired, tight bound we cannot simply apply Occam's Razor to Slice III since it is determined by roughly $2n \cdot K = 2n \cdot \epsilon k(n.\epsilon)$ bits.)

On the other hand, in order to learn a function in the class, we must learn Slice III with error at most 4ϵ. Using Item 1 of Proposition 3.1, we obtain $\text{ITSC}(\text{Slice III}; n, 4\epsilon) = \Omega(k(n, 4\epsilon))$, and $\text{ITSC}(n, \epsilon) = \Omega(k(n, 4\epsilon)) = \Omega(k(n, \epsilon))$ follows. ■

Lemma 3.2 *The concept class described above has* computational *sample complexity* $\text{CSC}(n, \epsilon) = \Theta(g(n, \epsilon) \cdot k(n, \epsilon))$.

We stress that this lemma, as well as all subsequent lemmas which refer to computational complexity lower bounds, holds provided Slice I is indeed determined by a pseudorandom function.

Proof: Using Item 2 of Proposition 3.1, we have that $\text{CSC}(\text{Slice III}; n, \epsilon) = O(k(n, \epsilon))$, and using $g(n, \epsilon) \geq 1$ we infer that Slice III can be efficiently learned with ϵ error given a sample of size $O(g(n, \epsilon) \cdot k(n, \epsilon))$. We next show that such a sample suffices for learning Slice I and Slice II as well. Since the information fields in the i^{th} block of Slice II have density bounded above by 2^{-i}, in order to learn Slice II with error at most ϵ, it suffices to learn the first $\ell \stackrel{\text{def}}{=} \lceil \log_2(4/\epsilon) \rceil$ with error at most $\epsilon/2$. However, such an approximation might not suffice for learning Slice I sufficiently well. Nonetheless, we next show that a sample of size $O(g(n, \epsilon) \cdot k(n, \epsilon))$ suffices for efficiently determining (*exactly*) the first ℓ seeds (residing in the first ℓ blocks of Slice II) and thus determining the first ℓ blocks of Slice I.

Let $i \leq \ell$ and consider the m information fields in the i^{th} block of the Seed Encoder. Suppose that for some constant c' (to be specified), we have $2c' \cdot (g(n, \epsilon) \cdot k(n, \epsilon))$ random examples. Then the expected number of examples residing in the information fields of the i^{th} block is

$$2c' \cdot (g(n, \epsilon) \cdot k(n, \epsilon)) \cdot \frac{m}{g(n, 2^{-i}) \cdot k(n, 2^{-i})} . \tag{6}$$

Since $g(n, \epsilon) \cdot k(n, \epsilon) = \Omega(g(n, 2^{-i})k(n, 2^{-i}))$, for a suitable constant c'' (the depends on c' and on the constants in the omega notation), this expected number is $2c''m$. With overwhelmingly high probability (i.e., $1 - \exp(-\Omega(m))$), there are at least $c''m$ examples in the information fields of the i^{th} block (for every $i \leq \ell$). We set c' so that c'' will be such that, with overwhelmingly high probability, such a sample will miss at most $c_{\text{err}}m$ of these fields, where c_{err} is the constant in Item 2 of Theorem 2.1 (e.g., $c'' = 2/c_{\text{err}}$ will suffice). Invoking Item 2 of Theorem 2.1 (for the special case in which there are no errors but part of the code-word is missing), we obtain the seed encoded in the i^{th} block of Slice II. Since the probability of failure on a particular block is negligible, with very high probability we obtain the seeds in all the first ℓ blocks of Slice II. This concludes the proof of the upper bound.

We now turn to the lower bound and let $i \stackrel{\text{def}}{=} \lfloor \log_2(1/4\epsilon) \rfloor$. Considering the i^{th} block of Slice I, we will show that too small a sample does not allow information regarding the i^{th} seed to be obtained from Slice II. This will lead to the failure of the computational bounded learner, since without such information the i^{th} block of Slice I looks totally random (to this learner). Specifically, let c_{sec} be the constant in Item 3 of Theorem 2.1, and let $c' = 2/c_{\text{sec}}$. Suppose the learner is given

$$c'' \cdot g(n, \epsilon) \cdot k(n, \epsilon) \quad < \quad c' \cdot g(n, 2^{-i})k(n, 2^{-i}) \tag{7}$$

random examples. (The constant c'' is such that the last inequality holds.) Then, using a Chernoff Bound we infer that, with overwhelmingly high probability, we will have at most

$$2c' \cdot (g(n, 2^{-i}) \cdot k(n, 2^{-i})) \cdot \frac{m}{g(n, 2^{-i}) \cdot k(n, 2^{-i})} \quad = \quad 2c'm \quad = \quad c_{\text{sec}}m \tag{8}$$

random examples in the information fields of the i^{th} block of Slice II. Invoking Item 3 of Theorem 2.1, this yields no information regarding the seed encoded in the i^{th} block of Slice II. Thus, the computationally bounded learner cannot predict any unseen point in the i^{th} block of Slice I better than at random. This means that its error is greater than allowed (i.e., $\geq 2^{-i-1} > \epsilon$). Thus, $\text{CSC}(n, \epsilon) > c'' \cdot g(n, \epsilon)k(n, \epsilon)$, where c'' is a constant as required. ■

4 The Two Equalizers (Proof of Theorem 1.6)

In this section we show that : (1) the sample equalizer (i.e. Slice III) described in Section 3.2 can be used to prove the first item in Theorem 1.6 (noisy-sample equalizer); (2) another construction, based on *interval functions* can be used to prove the second item of the theorem (query equalizer).

4.1 Noisy-Sample Equalizer (Item 1 of Theorem 1.6)

As noted above, we use Slice III of the construction in Section 3.2. Here we think of a concept in the class $\mathcal{S} = \cup_n \mathcal{S}_n$ as being an array of size 2^n (as opposed to 2^{n-2} when it serves as the third slice of a concept). The number of sublocks in each of the $t = n - O(\log(n))$ blocks is $p(n)$. The proof follows the structure of the proof of Proposition 3.1.

Lemma 4.1 $\text{ITSC}(\mathcal{S}, n, \epsilon, \gamma) = \Omega\left(\frac{p(n)}{\epsilon\gamma^2}\right)$.

Proof: Let $i \stackrel{\text{def}}{=} \lfloor \log_2(1/2\epsilon) \rfloor$. In order to learn a function in the class $\mathcal{S}_n$, one must learn the i^{th} block reasonably well. Specifically, one must obtain sufficiently many examples from the i^{th} block, or else the information we obtain on the bits residing in this block is too small. In particular, we claim that we must have at least $\Omega(p(n)/\gamma^2)$ examples in the i^{th} block. We prove the claim by noting that it corresponds to the classical Information Theoretic measure of the mutual information (cf., [CT91]) that these samples provide about the string residing in this block. Each example provides information on a single bit residing in the block and the amount of information is merely the capacity of a Binary Symmetric Channel with crossover probability $\eta = \frac{1}{2} - \gamma$. That is, each example yields $1 - H_2(\eta)$ bits of information, where H_2 is the binary entropy function, which satisfies $1 - H_2(0.5 - \gamma) \approx \Theta(\gamma^2)$. The claim follows by additivity of information.

Let $c > 0$ be a constant so that we must have at least $p(n)/c\gamma^2$ examples in the i^{th} block. Then, in order to have (with high probability) at least $p(n)/c\gamma^2$ examples in the i^{th} block, the total number of random samples must be at least $2^i \cdot \frac{p(n)}{2c\gamma^2} = \Omega\left(\frac{p(n)}{\epsilon\gamma^2}\right)$. ■

Lemma 4.2 $\text{CSC}(\mathcal{S}, n, \epsilon, \gamma) = O\left(\frac{p(n)}{\epsilon\gamma^2}\right)$.

Proof: We consider the fraction of a concept in S_n that can be correctly inferred by a sample of $O(p(n)/\epsilon\gamma^2)$ random examples. It suffices to show that the total area left incorrectly inferred in the first $\ell \stackrel{\text{def}}{=} \lceil \log_2(4/\epsilon) \rceil$ blocks is at most an $\epsilon/2$ fraction of the total domain (since the remaining blocks cover at most an $\epsilon/2$ fraction of the total). Fixing any $i \leq \ell$, we consider the expected number of sub-blocks incorrectly inferred in the i^{th} block (out of the total $p(n)$ sub-blocks). We use the obvious inference rule – a majority vote. Thus, a sub-block is correctly inferred if and only if a strict majority of the examples obtained from it are labeled correctly. (Having obtained examples from this sub-block is clearly a necessary condition for having a strict majority.) Using a Chernoff bound, the probability that we do not have the correct majority in $O(p(n)/\epsilon\gamma^2)$ trials is at most $\exp(-\Omega(2^{-i+1}/\epsilon))$. As in the proof of Proposition 3.1, it follows that the expected fraction of bits that are incorrectly inferred in the first ℓ blocks is bounded above by $\epsilon/20$ and the Lemma follows. ■

4.2 Query Equalizer (Item 2 of Theorem 1.6)

For every n, we consider the following concept class $\mathcal{S}_n$. Each concept in the class consists of $p(n)$ blocks of equal size $Q \stackrel{\rm def}{=} 2^{n-2}/p(n)$. Each block corresponds to an *interval* function. Namely, the bit locations in each block are associated with $[Q] \stackrel{\rm def}{=} \{1, ..., Q\}$, and the bits themselves are determined by a pair of integers in $[Q]$. If the $i^{\rm th}$ block is associated with a pair (u_i, v_i), then the $j^{\rm th}$ bit in this block is 1 if and only if $u_i \leq j \leq v_i$. Note that pairs (u_i, v_i) with $u_i > v_i$ determine an all-zero block.

Lemma 4.3 $\text{ITQC}(\mathcal{S}, n, \epsilon) = \Omega(p(n)/\epsilon)$.

Proof: We start by bounding the expected relative error of the algorithm on a single block when making at most q queries to this block. We later discuss the implication of such a bound on the total error on S_n. Suppose first that the learner is deterministic. Then, no matter how it chooses its q queries, there exists an interval of length $1/q$ which is never queried. Hence the algorithm cannot distinguish the case in which the target concept is all 0's and the case the target concept has 1's only in the non-queried interval, and must have error at least $1/2q$ on at least one of these concepts.

Next, we consider a probabilistic learner which makes q queries all of them are answered by 0 (as would be the case for the all-zero concept). Then, for every $\delta > 0$, there must exists an interval of relative length δ/q (i.e., actual length $\delta Q/q$) so that the probability that a query was made in this interval is below δ. We again consider the all-zero concept and the concept which has 1's only in this interval. We consider a $1-\delta$ fraction of the runs of the algorithm in which no query is made to the above interval. In these runs the algorithm cannot distinguish the all-zero concept from the other concept. Thus, with probability $\frac{1-\delta}{2}$ the algorithm has error at least $\delta/2q$ (on some concept). If we set $\delta = 0.2$ and $q = \frac{1}{100\epsilon}$, then we have that with probability at least 0.4 the error is at least 10ϵ on one of the two concepts.

To analyze the execution of a learning (with queries) algorithm on a (complete) concept in S_n, we consider the following game, consisting of two stages. In the *first stage*, the algorithm makes q queries in each block. The algorithm is not charged for any of these queries. In the *second stage*, the algorithm makes a choice, for each block, whether to output a hypothesis for this block or to ask for additional queries. In the latter case it is supplied with an infinite number of queries (for this block) and gets charged only for the q original queries made in the first stage. At the end of the second stage the algorithm must output a hypothesis for each of the remaining blocks. The algorithm is required to output hypotheses which together form an ϵ-approximation of the target. Clearly, the charges incurred in the above game provide a lower bound on the actual query complexity of any learning algorithm.

Claim: Any algorithm that learns S_n with ϵ error and confidence 0.9, incurs a charge of at least $0.04 \cdot p(n) \cdot q$.

Proof: Consider the following mental experiment in which an algorithm executes only the first stage, and all its queries are answered '0' (as if each block corresponds to the empty interval function). For each i, let I_i be an interval (of maximum length) in the $i^{\rm th}$ block such that the probability that a query is made (in the above mental experiment) to this interval is below δ (for $\delta = 0.2$). We next define a distribution on $2^{p(n)}$ possible target concepts: For the $i^{\rm th}$ block, independently, with probability 1/2, the interval I_i is chosen, and with probability 1/2, the empty interval is chosen. Now consider a full (two stage) execution of an algorithm that learns S_n with ϵ error and confidence 0.9, when the target is chosen according to the above distribution. Since the bound on the error and confidence of the algorithm are with respect to a worst-case choice of a target concept, it must still hold that with probability at least 0.9 over the randomization of the algorithm *and* the random choice of the target, the error of the algorithm is at most ϵ.

Suppose, towards contradiction, that this algorithm incurs charge less than $0.04p(n)q$ (where q is set as above). Then, for at least 0.96% of the blocks, the algorithm outputs a hypothesis at the end of the first stage. By our assumption on the algorithm, with probability at least 0.9, the overall error in these hypotheses must be bounded by ϵ, and so at most one ninth of these blocks may have *relative* error greater than $10\epsilon >$. But this implies that, with probability at least $0.9 \cdot 0.96 \cdot \frac{8}{9} > 0.6$, the algorithm has relative error smaller than 10ϵ on a randomly located block, in contradiction to the above analysis of the single-block case. ■

Lemma 4.4 $\mathrm{CSC}(\mathcal{S}, n, \epsilon) = O(p(n)/\epsilon)$.

Proof: The computationally bounded learner simply finds a minimal consistent hypothesis for each block in the concept. Namely, for the i^{th} block it lets $\hat{u}_i \in [Q]$ be the smallest index of an example labeled 1 that belongs to i^{th} block, and it lets $\hat{v}_i \in [Q]$ be the largest index of an example labeled 1 in the i^{th} block. If no example in the block is labeled 1, then it lets $\hat{u}_i = [Q]$, and $\hat{v}_i = 1$ (so the hypothesis is all 0).

Assume the learner is given a sample of size $bp(n)/\epsilon$ $(= b \cdot k(n, \epsilon))$ for some constant $b > 1$. Then, for any particular block, the expected number of examples that fall in the block is b/ϵ, and the probability that less than $b/(2\epsilon)$ belong to the block is $\exp(-\Omega(b/\epsilon))$. Thus, by Markov's inequality, for sufficiently large b, the probability that the fraction of blocks receiving less than $b/2\epsilon$ examples exceeds $\epsilon/2$, is a small constant. It remains to show that with high probability the total error in the blocks receiving a sufficient number of examples is at most $\epsilon/2$. To this end we show that for each such block, the expected error, relative to the size of the block, is at most ϵ/b' for some constant b'.

Consider a particular block that receives at least $s = b/(2\epsilon)$ examples. Let the interval defining the block be $[u, v]$, and let the hypothesis of the learner be $[\hat{u}, \hat{v}]$. First note that by definition of the algorithm, $[\hat{u}, \hat{v}]$ is always a subinterval of $[u, v]$, and hence we have only one-sided error. In particular, in the case that $u > v$ (i.e., the target interval is empty), the error of the hypothesis is 0. Thus assume $u \leq v$. For sake of the analysis, if $\hat{u} > \hat{v}$ (i.e., the learner did not observe any positive example in the block, and the hypothesis is all 0), redefine $\hat{u}$ to be $v + 1$, and $\hat{v}$ to be v. By definition, the hypothesis remains all 0. Let $\zeta_{\mathrm{L}} \stackrel{\text{def}}{=} (\hat{u} - u)/Q$, and $\zeta_{\mathrm{R}} \stackrel{\text{def}}{=} (v - \hat{v})/Q$. The error of the hypothesis (relative to the block) is the sum of these two random variables. We next bound the expected value of ζ_{L} (the expected value of ζ_{R} is bounded analogously).

For any integer a, the probability that $\zeta_{\mathrm{L}} > a \cdot \frac{1}{s}$ is the probability that no example fell between u and $u + (a/s) \cdot Q$, which is $(1 - a/s)^s < \exp(-a)$. Therefore,

$$\begin{aligned}
\mathrm{Exp}(\zeta_{\mathrm{L}}) &< \sum_{a=0}^{s-1} \Pr\left(\frac{a}{s} < \zeta_{\mathrm{L}} \leq \frac{a+1}{s}\right) \cdot \frac{a+1}{s} & (9) \\
&< \sum_{a=0}^{s-1} \Pr\left(\zeta_{\mathrm{L}} > \frac{a}{s}\right) \cdot \frac{a+1}{s} & (10) \\
&< \frac{1}{s} \sum_{a=0}^{\infty} (a+1) e^{-a} & (11)
\end{aligned}$$

$$< \; 3/s \; = \; 6\epsilon/d \; . \tag{12}$$

Thus, the (total) expected error of the algorithm on all blocks that receive at least $s = b/(2\epsilon)$ examples, is bounded by $12\epsilon/b$. ■

5 The General Construction

We adopt the structure of the construction presented in Section 3. Specifically, the Pseudorandom Slice remains the same here, and the Sample Equalizer is one of the two equalizers analyzed in Section 4 (depending on the application). The main modification is in the Seed Encoder Slice (Slice II). For an illustration of the construction, see Figure 2.

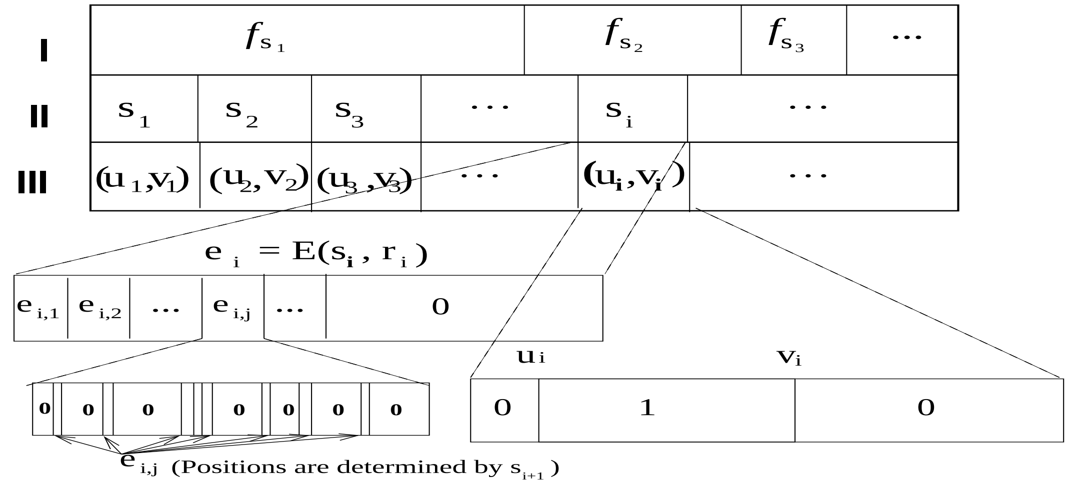

Figure 2: The General Construction with the Query Equalizer (analyzed in Item 2 of Theorem 1.6)

As in Section 3, we encode each seed using the probabilistic coding scheme of Theorem 2.1. In Section 3 we repeated each resulting bit (of each encoded seed) in each bit of a corresponding information field, where the information fields occupied only a small fraction of Slice II and their locations were fixed. Here we augment this strategy by having only few of the bits of these information fields equal the corresponding bit and the rest be set to zero. Thus, only few locations in each information fields are really informative. Furthermore, the locations of the informative bits will not be known a-priori but will rather be "random" (as far as the computational bounded learner is concerned). This strategy effects the computational bounded learner both in the query and noisy models. Using queries the computational bounded learner may focus on the informative fields but it cannot hit informative bits within these fields any better than at random.

Suppose that a ρ fraction of the bits in an information field are really informative. Then to distinguish a '0' from a '1' (with constant confidence) the learner must make $\Theta(1/\rho)$ queries/examples into the information field. Things become even harder in the presence of classification noise at rate $\eta = 0.5 - \gamma$, say, greater than 0.2. In this case, when getting an example in the information field, or even making a query to the information field, the answer may be '0' both in case the bit is not informative (and the answer is correct) and in case the bit is informative. The latter case happens with probability $0.5 + \gamma$ (resp., $0.5 - \gamma$) when the real information is '0' (resp., '1'). Thus, in case the information bit is '0' (resp., '1'), the answer is '0' with probability $1 - \eta$ (resp., with probability $(1-\rho)\cdot(0.5+\gamma) + \rho\cdot(0.5-\gamma) = 1 - \eta - 2\rho\gamma$). As $1 - \eta$ is bounded away from (both 0 and)

1, distinguishing an encoding of '0' from an encoding of '1' (with constant confidence) requires $\Theta(1/(\rho\gamma)^2)$ queries/examples into the information field.

The above discussion avoids the question of how we can make the informative locations be "random" (as far as the computational bounded learner is concerned). These "random" locations must be part of the specification of the concepts in the class. We cannot have truly random locations if we want to maintain polynomial-size description of individual concepts. The use of pseudorandom functions is indeed a natural solution. There is still a problem to be resolved – the computational bounded learner must be able to obtain the locations of "information" for those blocks that it needs to learn. Our solution is to use the $i+1^{\text{st}}$ pseudorandom function in order to specify the locations in which information regarding the i^{th} seed is given.

The following description is in terms of two density functions, denoted $\rho_1, \rho_2 : \mathcal{N} \times \mathcal{R} \mapsto \mathcal{R}$. Various instantiations of these functions will yield all the results in the paper. Each function in the concept class consists of the three slices, and each slice is further sub-partitioned into *blocks* as described below.

SLICE I: THE PSEUDORANDOM PART. Exactly as in Section 3. That is, the i^{th} block has size 2^{n-1-i} and is determined by a pseudorandom function $f_{s_i} : \{0,1\}^{n-1-i} \to \{0,1\}$, where the seed s_i is of length n.

SLICE II: THE SEEDS ENCODER. As in Section 3, this slice is partitioned into t blocks (of equal size) so that the i^{th} block has size $B \stackrel{\text{def}}{=} \frac{1}{t} \cdot 2^{n-2}$ and encodes the i^{th} seed, s_i, using an additional string r_i of length $O(n)$. Let e_i be the codeword obtained by employing the probabilistic encoding scheme on input s_i using randomness r_i. Recall that $m \stackrel{\text{def}}{=} |e_i| = O(n)$. The i^{th} block is further divided into m (consecutive) *information fields*, each of size $\rho_1(n, 2^{-i}) \cdot \frac{B}{m}$, and an additional *empty field* (of size $(1 - \rho_1(n, 2^{-i}) \cdot B)$.

A $\rho_2(n, 2^{-i})$ fraction of the bits in the j^{th} information field are set to the value of the j^{th} bit of e_i. These bits are called *informative* and their locations are determined ("randomly") by a pseudorandom function, $h_{s_{i+1}} : \{0,1\}^n \mapsto \{0,1\}^n$ (this function uses the $i+1^{\text{st}}$ seed!). The remaining bits in each information field as well as all bits of the empty field are set to zero.

A few details are to be specified. Firstly, bit locations in Slice II are associated with strings of length $n-2$. Thus, bit location $\alpha \in \{0,1\}^{n-2}$ that is inside an information field is informative if and only if $h_{s_{i+1}}(11\alpha)$ is among the first $2^n \cdot \rho_2(n, 2^{-i})$ strings in the lexicographic order of all n-bit long strings. The pseudorandom functions (over the domain $\{0,1\}^{n-i-1}$) used in Slice I are determined by the same seeds by letting $f_{s_i}(z) = \text{LSB}(h_{s_i}(0^{i+1}z))$, where $\text{LSB}(\sigma_n \cdots \sigma_1) = \sigma_1$ is the least significant bit of $\sigma_n \cdots \sigma_1$. Thus, there is no "computational observable" interference between the randomness used for determining informative locations and the randomness used in Slice I.

Note that Slice II in the construction of Section 3 is obtained by setting $\rho_1(n, \epsilon) = \frac{tm}{g(n,\epsilon)k(n,\epsilon)}$ and $\rho_2 \equiv 1$.

SLICE III: THE EQUALIZER. In this Slice we use one of the two equalizers analyzed in Theorem 1.6 (depending on the application).

5.1 Analysis of Slices I and II.

It will be convenient to analyze the concept class that results from the above by omitting Slice III (the Equalizer). We refer to the resulting class as to the core class.

Fact 5.1 *The core class has information theoretic sample complexity* $\text{ITSC}(n, \epsilon, \gamma) = O(n^2/\epsilon\gamma^2) = O(k(n, \epsilon)/\gamma^2)$.

Proof: Follows from the Noisy Occam Razor [Lai88]. ■

We are not interested in an information theoretic lower bound for the core class since the Equalizer will dominate the information theoretic complexities. Thus, we turn to analyze the computational complexities of the core class.

Lemma 5.2 *The core class has*

1. *computational* (noiseless) sample *complexity* $\mathrm{CSC}(n,\epsilon) = \Theta(\frac{n^2}{\rho_1(n,\epsilon)\cdot\rho_2(n,\epsilon)})$, *provided that* $\frac{1}{(\rho_1\cdot\rho_2)}$ *is an admissible function and that it is lower bounded by* $1/\epsilon$.
2. *for* $\gamma \leq \frac{1}{4}$, *computational* (noisy) sample *complexity* $\mathrm{CSC}(n,\epsilon,\gamma) = \Theta(\frac{n^2}{\rho_1(n,\epsilon)\cdot\rho_2(n,\epsilon)^2\cdot\gamma^2})$, *provided that* $\frac{1}{\rho_1\cdot\rho_2^2}$ *is an admissible function and that it is lower bounded by* $1/\epsilon$.
3. *computational* query *complexity* $\mathrm{CQC}(n,\epsilon) = \Theta(\frac{n}{\rho_2(n,\epsilon)})$, *provided that* $\frac{1}{\rho_2}$ *is an admissible function and that it is lower bounded by* $1/\epsilon$.

Proof: *Item 1* (noise-less sample complexity:) This item follows by observing that arguments used in Lemma 3.2 can be modified to obtain the desired bound. Consider the i^{th} block of Slice II. We first note that a random example hits an information field of the i^{th} block with probability $\rho_1(n,2^{-i})/t$ (i.e., with probability $1/t$ it falls in the i^{th} block and conditioned on being in the i^{th} block it falls in an information field with probability $\rho_1(n,2^{-i})$). Thus, the probability of hitting a specific information field (out of the $m=\Theta(n)$ fields) is $\frac{\rho_1(n,2^{-i})}{tm} = \Theta(\frac{\rho_1(n,2^{-i})}{n^2})$. We also know that a random example in an information field is informative (i.e., depends on the encoded bit) with probability $\rho_2(n,2^{-i})$ and is set to zero otherwise.

For the lower bound, consider the i^{th} block for $i \stackrel{\text{def}}{=} \lfloor\log(1/4\epsilon)\rfloor$. Similarly to what was show in the lower-bound of Lemma 3.2, for an appropriate constant c'', a sample of size $c''\cdot\frac{n^2}{(\rho_1(n,\epsilon)\cdot\rho_2(n,\epsilon))}$ will contain informative examples in less than $c_{\text{sec}}\,m$ of the information fields in the i^{th} block (where c_{sec} is the constant in Item 3 of Theorem 2.1). As a result, the i^{th} seed cannot be obtained and the error of the learner on Slice I is too large.

The proof of the upper bound is easily adapted as well. As in the proof of Lemma 3.2 we have that for a sufficiently large constant c', with very high probability, a sample of size $c'\cdot n^2/(\rho_1(n,\epsilon)\cdot\rho_2(n,\epsilon))$ will contain at least one informative example in all but a small constant fraction of the m information fields in the i^{th} block, for every $i \leq \ell \stackrel{\text{def}}{=} \lceil\log(8/\epsilon)\rceil$. The only difference here is that while seeing a '1' in an information field in fact means that the bit encoded in it is '1', seeing only '0's in the field only provides statistical evidence towards '0'. Thus, while in Lemma 3.2 we only had to deal with missing informative examples (that encode seeds), here we might have errors when inferring that the bit encoded is '0'. However, the coding scheme (of Theorem 2.1) allows a constant fraction (i.e., c_{err}) of errors, and hence we can handle a constant fraction of errors in each of the first ℓ blocks. Note that the ℓ corresponding seeds determine not only the first ℓ blocks of Slice I but also the *locations* of informative bits in the first $\ell-1$ blocks of Slice II.

Item 2 (noisy sample complexity:) The additional difficulty we encounter here (as compared to Item 1 above) is that due to the noise it does not suffice to "hit" informative examples inside information fields in order to infer the encoded bit. Namely, each example (informative or not) has an incorrect label with probability $\eta = \frac{1}{2}-\gamma$. Therefore, seeing a '1' in an information field does not necessarily mean that it is an informative example and the field encodes the bit '1', but rather it could be a noisy bit in an information field encoding the bit '0'.

However, there is clearly still a difference between information fields that encode '1', and those that encode '0': In case an information field encodes '0', a random example in it will be labeled '1' with probability η. On the other hand, in case an information field encodes '1', a random example in it will be labeled '1' with probability $\rho_2(n, 2^{-i}) \cdot (1-\eta) + (1-\rho_2(n, 2^{-i})) \cdot \eta = \eta + 2\gamma\rho_2(n, 2^{-i})$. Thus, we need to distinguish a 0-1 sample with expectation η from a 0-1 sample with expectation $\eta + \Theta(\gamma\,\rho_2(n, 2^{-i}))$. This is feasible (with high probability) using $O(1/(\gamma^2\rho_2(n, 2^{-i}))$ examples. Since our coding scheme can suffer a constant fraction of errors, we can allow that a small fraction of information field will receive less than the required number of examples, and that among those receiving the desired number, a small fraction will be determined incorrectly. The upper bound follows.

For η that is bounded below by some constant (say, $\eta \geq 1/4$), a sample of size $\Omega(1/(\rho_2(n, 2^{-i})\gamma)^2)$ is also required to distinguish between the two cases discussed above with probability greater than, say $1/2 + c_{\text{sec}}/8$, where c_{sec} is the constant defined in Item 3 of Theorem 2.1. Suppose that a sample of size $c'm/(\rho_2(n, 2^{-i})\gamma)^2$ falls in the i^{th} block of Slice II. Then, for sufficiently small c', at most $c_{\text{sec}}/2$ information fields receive a sufficient number of examples. Hence, even if all these information fields are correctly inferred, by Remark 2.3, with very high probability no information about the ith seed will be revealed. In particular, this holds for $i = \lfloor \log_2(1/4\epsilon) \rfloor$. We conclude that, for $\gamma \leq 1/4$,

$$\text{CSC}(n, \epsilon, \gamma) = \Theta\left(\frac{n^2}{\rho_1(n, \epsilon)} \cdot (\rho_2(n, \epsilon) \cdot \gamma)^{-2}\right) \tag{13}$$

Item 3. Using similar arguments to those applied above we show that $O(n/\rho_2(n, \epsilon))$ queries suffice for efficiently determining the first $\ell \stackrel{\text{def}}{=} \lceil \log_2(8/\epsilon) \rceil$ seeds residing in the first ℓ blocks of Slice II. Specifically, we use a $2^{-\ell-4+i}$ fraction of the $c' \cdot (n/\rho_2(n, \epsilon))$ queries as a random sample into the information fields of i^{th} block, for $i = 1, ..., \ell$ (and ignore the empty fields in all blocks). Thus, we have $c' \cdot 2^{-\ell-4+i} \cdot (n/\rho_2(n, \epsilon)) = 2c'' \cdot m/\rho_2(n, 2^{-i})$ random examples in the i^{th} block, where c'' is a constant related to the constant c'. With overwhelmingly high probability we'll obtain at least $c''m$ informative bits. For a suitable choice of the constants (c' and c''), this suffices to recover the i^{th} seed for every $i \leq \ell$. Observe that the only part in which we have used queries is in the skewing of the random examples among the various blocks.

We now turn to the lower bound and let $i \stackrel{\text{def}}{=} \lfloor \log_2(1/4\epsilon) \rfloor$. Considering the i^{th} block of Slice I, we show that, as long as the informative locations in the i^{th} block of Slice II are unknown, too few queries do not allow to obtain information regarding the i^{th} seed. This will lead to the failure of the computational bounded learner, since without such information the i^{th} block of Slice I looks totally random (to this learner). The actual argument starts from the last block (i.e., t^{th} block) and proceeds up to the i^{th} block. Assuming that the learner has no knowledge of the j^{th} seed, for $j > i$, we show that he obtains no knowledge of the $j - 1^{\text{st}}$ seed. On top of what is done in the analogous part of the proof of Lemma 3.2, we need to argue that having no knowledge of the j^{th} seed puts the learner in the same situation as if it has selected its queries at random: We can think of it making a query and then having a random biased coin determine if this query (into the Seed Encoder) carries information. ■

5.2 Applications of Lemma 5.2

Proof of Theorem 1.2. As in the setting for Theorem 1.1, we set $\rho_1(n, \epsilon) \stackrel{\text{def}}{=} \frac{n^2}{g(n,\epsilon) \cdot k(n,\epsilon)}$, and $\rho_2(n, \epsilon) \stackrel{\text{def}}{=} 1$, where $k(n, \epsilon) \stackrel{\text{def}}{=} \gamma^2 \cdot k(n, \epsilon, \gamma)$. By Item 1 of Theorem 1.6 and Fact 5.1, we have

$\text{ITSC}(n,\epsilon,\gamma) = \Theta(k(n,\epsilon,\gamma)$. Invoking Item 2 of Lemma 5.2 for $\gamma \leq 1/4$, and Item 1 of Lemma 5.2 for $\gamma > 1/4$ (i.e., constant γ), we have $\text{CSC}(n,\epsilon,\gamma) = \Theta(g(n,\epsilon)\cdot k(n,\epsilon,\gamma)$.

Proof of Theorem 1.3. The first item follows by setting $\rho_1(n,\epsilon) \stackrel{\text{def}}{=} \frac{n^2}{g_1(n,\epsilon)\cdot k(n,\epsilon)}$, and $\rho_2(n,\epsilon) \stackrel{\text{def}}{=} \frac{1}{g_2(n,\epsilon)}$. By Item 1 of Theorem 1.6 and Fact 5.1, we have $\text{ITSC}(n,\epsilon,\gamma) = \Theta(k(n,\epsilon)/\gamma^2)$. Invoking Item 1 of Lemma 5.2, we have $\text{CSC}(n,\epsilon) = \Theta(g_1(n,\epsilon)g_2(n,\epsilon)\cdot k(n,\epsilon))$. Invoking Item 2 of Lemma 5.2, we have $\text{CSC}(n,\epsilon,\gamma) = \Theta(g_1(n,\epsilon)g_2(n,\epsilon)^2\cdot k(n,\epsilon)/\gamma^2)$, for every $\gamma \leq 1/4$.

The second item follows by setting $\rho_1(n,\epsilon) \stackrel{\text{def}}{=} 1$, and $\rho_2(n,\epsilon) \stackrel{\text{def}}{=} \frac{n^2}{k(n,\epsilon)}$. Again, we have $\text{ITSC}(n,\epsilon,\gamma) = \Theta(k(n,\epsilon)/\gamma^2)$, and invoking Items 1 and 2 of Lemma 5.2, we have $\text{CSC}(n,\epsilon) = \Theta(k(n,\epsilon))$ and $\text{CSC}(n,\epsilon,\gamma) = \Theta(k(n,\epsilon)^2/n^2\gamma^2))$, for every $\gamma \leq 1/4$. Actually, we can obtain a more general result by setting $\rho_1(n,\epsilon) \stackrel{\text{def}}{=} n^{-(d-a-2)}\epsilon^{1-\beta}$ and $\rho_2(n,\epsilon) \stackrel{\text{def}}{=} n^{-d}\epsilon^{\beta}$, for any $a \in [0, d-2]$ and $\beta \in [0,1]$.

Proof of Theorem 1.4, for $g_2 \geq n$. Here we set $\rho_1(n,\epsilon) \stackrel{\text{def}}{=} \frac{n}{g_2(n,\epsilon)}$, and $\rho_2(n,\epsilon) \stackrel{\text{def}}{=} \frac{n}{g_1(n,\epsilon)\cdot k(n,\epsilon)}$. (The hypothesis $g_2(n,\epsilon) \geq n$ guarantees that $\rho_1(n,\epsilon) \leq 1$ as required by the admissibility condition.) By Item 2 of Theorem 1.6 and Fact 5.1, we have $\text{ITQC}(n,\epsilon) = \Theta(k(n,\epsilon)) = \Theta(\text{ITSC}(n,\epsilon))$. Invoking Item 1 of Lemma 5.2, we have $\text{CSC}(n,\epsilon) = \Theta(g_1(n,\epsilon)g_2(n,\epsilon)\cdot k(n,\epsilon))$. Invoking Item 3 of Lemma 5.2, we have $\text{CQC}(n,\epsilon) = \Theta(g_1(n,\epsilon)\cdot k(n,\epsilon))$.

5.3 Proof of Theorem 1.4 (for arbitrary g_2).

The core class (analyzed in Lemma 5.2) provides a computationally bounded learner that uses queries an advantage over a computationally bounded learner that only uses uniformly distributed examples. Whereas the former may focus its queries on the first $\log_2(2/\epsilon)$ blocks of Slice II, the latter may not. Thus, typically, using queries entitles an advantage of a factor of $n/\log(1/\epsilon)$ in trying to learn Slice II above. To close this gap (and allow to establish Theorem 1.4 for arbitrary g_2), we modify Slice II as follows. The basic idea is to randomly permute the locations of the information fields of the various blocks. Thus the query-learner is forced to look for the bits it needs in all possible locations (rather than "zoom-in" on the appropriate block).

Slice II (modified). Let $t \stackrel{\text{def}}{=} n/\log_2 n$ (rather than $t = n - \log_2 n$). This slice is partitioned into $t\cdot m$ fields of equal size, $F \stackrel{\text{def}}{=} \frac{1}{tm}\cdot 2^{n-2}$, where m is (as before) the length of the encoding of an n-bit long seed. Unlike the above construction, we do not have a common empty field (instead each field contains an informative part and an empty part as described below). We use m permutations over $\{1,...,t\}$, denoted $\pi_1,...,\pi_m$, to determine the correspondence between fields and seed-information. Specifically, the j^{th} bit of the i^{th} seed is "encoded" in field number $(j-1)\cdot t + \pi_j(i)$. (The permutations are part of the description of the concept.) Each field corresponding to one of the bits of the i^{th} seed consists of two parts. The first part, containing the first $\rho_1(n, 2^{-i})\cdot F$ bits of the field, carries information about the corresponding bit of the seed; whereas the second part (the rest of the field's bits) is uncorrelated to the seed. Loosely speaking, the *informative part* contains the results of independent coin flips each with bias $\rho_2(n,2^{-i})$ towards the correct value of the corresponding bit (i.e., the probability that the answer is correct is $0.5 +$ bias); whereas the rest contains the results of independent unbiased coin flips. Actually, the random choices are implemented by a pseudorandom function determined by the $i+1^{\text{st}}$ seed.

Lemma 5.3 *Assume that $\frac{1}{\rho_1}$ and $\frac{\epsilon}{\rho_2^2}$ are admissible functions. Then the modified core class has*

1. *computational* (noise-less) sample *complexity* $\text{CSC}(n,\epsilon) = \Theta(\frac{n^2}{\rho_1(n,\epsilon)\cdot\rho_2(n,\epsilon)^2})$.

2. *computational* query *complexity* $\text{CQC}(n, \epsilon) = \Theta(\frac{n^2}{\rho_2(n,\epsilon)^2})$.

Proof: We follow the structure of the proof of Lemma 5.2, indicating the necessary modifications.

Item 1. Considering Slice II, we note that a random example hits an information part of a field belonging to the i^{th} seed with probability $\rho_1(n, 2^{-i})/t$. Intuitively, $\Theta(\rho_2(n, 2^{-i})^{-2})$ such hits are required for obtaining reliable information from this field.

For the lower bound, we assume that the learner is given the permutations π_j for free. Still, the arguments used in Lemma 5.2 imply that it needs $\Omega(\rho_2(n, 2^{-i})^{-2} \cdot m)$ hits in the fields of the i^{th} seed in order to recover this seed. Using $t, m = \Omega(n)$, the lower bound follows.

The proof of the upper bound is to be adapted as here we cannot assume that the permutations π_j are known to the learner. For $i = 1, ..., \log_2(8/\epsilon)$, the learner determines the i^{th} seed as follows. For $j = 1, ..., m$, the learner determines the value of the j^{th} bit in the encoding of the i^{th} seed. It considers only examples in the $\rho_1(n, 2^{-i}) \cdot F$ prefix of each of the relevant fields; that is, fields with indices $(j-1) \cdot t + 1, ..., (j-1) \cdot t + t$. For each such field it estimates the bias of the field. With high constant probability, the estimated bias of field $(j-1) \cdot m + \pi_j(i)$ is approximately $\rho_2(n, 2^{-i})$ and, under our assumption on ρ_2, every other field corresponding to the j^{th} bit of an encoding of some other seed, has significnatly different bias in its $\rho_1(n, 2^{-i} \cdot F$ prefix. Thus, this bit is obtained correctly with high constant probability. As usual, this allows to decode correctly the entire seed. Having the i^{th} seed we may determine the i^{th} pseudorandom function as well as the "encoding" of the bits of the $i - 1^{\text{st}}$ codeword (i.e., one may efficiently determine the value of each bit in each of the fields corresponding to the $i - 1^{\text{st}}$ seed). The upper bound follows.

Item 2. The upper bound follows easily by the obvious adaptation of the above learning strategy (i.e., when trying to determine the j^{th} bit in the encoding of the i^{th} seed the learner makes queries only to the $\rho_1(n, 2^{-i}) \cdot F$ prefix of each of the relevant fields).

For the lower bound we need to modify the argument given above (as here we cannot afford giving away the permutations π_j for free). In fact, the whole point of the modification was to deprive the learner from such information. Still, when arguing about the i^{th} seed, for $i = \log_2(4/\epsilon)$, we may give the learner $\pi_j(1), ..., \pi_j(i-1)$ $(\forall j)$ for free. We may assume without loss of generality that the learner does not make queries to these fields (i.e., to field with index $(j-1) \cdot t + \pi_j(i')$ for $i' < i$ and any j). Based on our encoding scheme, the learner must infer $\Omega(m)$ of the bits in the encoding of the i^{th} seed. For each bit it must discover $\pi_j(i)$ and make $\Omega(\frac{1}{\rho_2(n,2^{-i})})$ queries. However, for every j and $i' \geq i$, the field corresponding to the j^{th} bit of the encoding of the i'th seed has bias (in its $\rho_1(n, 2^{-i}) \cdot F$ prefix) that is bounded above by $\rho_2(n, 2^{-i})$. Hence, for each j, the learner must perform $\Omega(n/(\rho_2(n, 2^{-i})^2)$ queries to these fields in order to infer the desired bit. The lower bound follows. ■

Theorem 1.4 (in its general form) now follows by setting $\rho_1(n, \epsilon) \stackrel{\text{def}}{=} \frac{1}{g_2(n,\epsilon)}$ and $\rho_2(n, \epsilon) \stackrel{\text{def}}{=} \sqrt{\frac{n^2}{g_1(n,\epsilon) \cdot k(n,\epsilon)}}$. We need $t = O(n/\log n)$ so that the additional "concept complexity" (of $\log_2((t!)^m)$) is dominated by $k(n, \epsilon)$ (the desired information-theoretic sample complexity). Alternatively, the theorem will hold provided $k(n, \epsilon) = \Omega(n^2 \log n)$.

6 Proof of Theorem 1.5

Here we merely use a pseudorandom generator [BM84, Yao82]. Specifically, we need a generator, G, which stretches seeds of length n into sequences of length $p(n)$. The concept class, $\mathcal{C} = \{C_n\}$, will correspond to all possible choices of a seed for the generator. Specifically, for every seed $s \in \{0,1\}^n$, we get a concept $f_s \in C_n$ defined so that $f_s(x) \stackrel{\text{def}}{=} \sigma_i$, where σ_i is the i^{th} bit of $G(s)$ and x belongs to the i^{th} subset in a "nice" $p(n)$-way partition of $\{0,1\}^n$ (e.g., a partition by lexicographic order in which all parts are of about the same size).

By Occam's Razor [BEHW87], the above concept class has $\mathcal{ITSC}(n,\epsilon) = O(n/\epsilon)$. On the other hand, considering a variation of the standard lower-bound distribution (i.e., which assigns probability $1-4\epsilon$ uniformly to all instances in a single subset and is uniform on all other instances), we derive the computational lower bound. Specifically, using $0.1 \cdot p(n)/\epsilon$ samples, with overwhelmingly high probability, the learner only sees $p(n)/2$ different bits of the pseudorandom sequence. As far as a computationally bounded learner is concerned, the rest of the sequence is random and so it will fail with probability at least $\frac{1}{2}$ when trying to predict any example corresponding to an unseen bit. Thus, $\mathcal{CSC}(n,\epsilon) > 0.1 \cdot p(n)/\epsilon$.

It is not hard to see that $10 \cdot p(n)/\epsilon$ samples allow efficient learning up to error ϵ (with respect to any distribution) and so $\mathcal{CSC}(n,\epsilon) \leq 10 \cdot p(n)/\epsilon$.

Acknowledgments

We are grateful to Moni Naor and Ronny Roth for helpful discussions.

References

[AD96] J. Aslam and S. Decatur. On the sample complexity of noise-tolerant learning. *Information Processing Letters*, 57:189–195, 1996.

[AK91] D. Angluin and M. Kharitonov. When won't membership queries help? In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, pages 444–453, 1991.

[Ang87] D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.

[BBCM95] C.H. Bennett, G. Brassard, C. Crépeau, and U. Maurer. Generalized privacy amplification. *IEEE Transaction on Information Theory*, 41(6):1915–1923, Nov. 1995.

[BEHW87] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Occam's razor. *Information Processing Letters*, 24(6):377–380, April 1987.

[BEHW89] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–865, 1989.

[BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13:850–864, 1984. Preliminary version in *23rd FOCS*, 1982.

[BW86] E. Berlekamp and L. Welch. Error correction of algebraic block codes. US Patent 4,633,470, 1986.

[CK78] I. Csiszár and J. Körner. Broadcast channels with confidential messages. *IEEE Transaction on Information Theory*, 24:339–348, 1978.

[CM97] C. Cachin and U.M. Maurer. Linking information reconciliation and privacy amplification. *Journal of Cryptology*, 10(2):97–110, 1997.

[CT91] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* Wiley, 1991.

[EHKV89] A. Ehrenfeucht, D. Haussler, M. Kearns, and L. Valiant. A general lower bound on the number of examples needed for learning. *Information and Computation*, 82(3):247–251, September 1989.

[GGM86] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Jour. of the ACM*, 33(4):792–807, Oct. 1986.

[HILL] J. Håstad, R. Impagliazzo, L.A. Levin, and M. Luby. Construction of pseudorandom generator from any one-way function. To appear in *SIAM J. on Computing*. Preliminary versions by Impagliazzo et. al. in *21st STOC* (1989) and Håstad in *22nd STOC* (1990).

[Jus72] J. Justesen. A class of constructive asymptotically good alegbraic codes. *IEEE Trans. Inform. Theory*, 18:652–656, 1972.

[Kha93] M. Kharitonov. Cryptographic hardness of distribution-specific learning. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 372–381, 1993.

[KV94] M. J. Kearns and L. G. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the Association for Computing Machinery*, 41:67–95, 1994. An extended abstract of this paper appeared in STOC89.

[Lai88] P. D. Laird. *Learning from Good and Bad Data.* Kluwer international series in engineering and computer science. Kluwer Academic Publishers, Boston, 1988.

[Mau91] U. M. Maurer. Perfect cryptographic security from partially independent channels. In *23rd STOC*, pages 561–571, 1991.

[NR95] M. Naor and O. Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. In *36th FOCS*, pages 170–181, 1995. Full version available from the ECCC at `http://www.eccc.uni-trier.de/eccc/`.

[PV88] L. Pitt and L. Valiant. Computational limitations of learning from examples. *Journal of the A.C.M.*, 35(4):965–984, 1988.

[Sim93] H. U. Simon. General bounds on the number of examples needed for learning probabilistic concepts. In *Proceedings of the Sixth Annual ACM Workshop on Computational Learning Theory*, pages 402–411. ACM Press, 1993.

[Tal94] M. Talagrand. Sharper bounds for Gaussian and empirical processes. *Ann. Probab.*, 22(1):28–76, 1994.

[Val84] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

[VC71] V.N. Vapnik and A.Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theor. Probability Appl.*, 16(2):264–280, 1971.

[Wyn75] A. D. Wyner. The wire-tap channel. *Bell System Technical Journal*, 54(8):1355–1387, Oct. 1975.

[Yao82] A. C. Yao. Theory and application of trapdoor functions. In *23rd FOCS*, pages 80–91, 1982.