Efficiency Improvements in Constructing Pseudorandom Generators from One-Way Functions

Electronic Colloquium on Computational Complexity, Report No. 89 (2010)

Efficiency Improvements in Constructing Pseudorandom Generators from One-way Functions

Iftach Haitner[*] and Omer Reingold[†] and Salil Vadhan[‡]

May 26, 2010

Abstract

We give a new construction of pseudorandom generators from any one-way function. The construction achieves better parameters and is simpler than that given in the seminal work of Håstad, Impagliazzo, Levin, and Luby [SICOMP '99]. The key to our construction is a new notion of *next-block pseudoentropy*, which is inspired by the notion of "inaccessible entropy" recently introduced in [Haitner, Reingold, Vadhan, and Wee, STOC '09]. An additional advantage over previous constructions is that our pseudorandom generators are parallelizable and invoke the one-way function in a non-adaptive manner. Using [Applebaum, Ishai, and Kushilevitz, SICOMP '06], this implies the existence of pseudorandom generators in NC^0 based on the existence of one-way functions in NC^1.

Keywords: one-way function; pseudorandom generator; pseudoentropy

1 Introduction

The result of Håstad, Impagliazzo, Levin, and Luby [14] that one-way functions imply pseudorandom generators is one of the centerpieces of the foundations of cryptography and the theory of pseudorandomness.

From the perspective of cryptography, it shows that a very powerful and useful cryptographic primitive (namely, pseudorandom generators) can be constructed from the minimal assumption for complexity-based cryptography (namely, one-way functions). With this starting point, numerous other cryptographic primitives can also be constructed from one-way functions, such as private-key cryptography [5, 21], bit-commitment schemes [22], zero-knowledge proofs for NP [6], and identification schemes [3].

From the perspective of pseudorandomness, it provides strong evidence that pseudorandom bits can be generated very efficiently, with smaller computational resources than the "distinguishers" to whom the bits should look random. Such kinds of pseudorandom generators are needed, for example, for hardness results in learning [27] and the natural proofs barrier for circuit lower bounds [23].

[*] Microsoft Research, New England Campus. E-mail: iftach@microsoft.com.

[†] Microsoft Research, Silicon Valley Campus, and Weizmann Institute of Science. E-mail:omreing@microsoft.com. Supported by US-Israel BSF grant 2006060.

[‡] School of Engineering and Applied Sciences and Center for Research on Computation and Society, Harvard University. E-mail: salil@seas.harvard.edu. Supported by NSF grant CNS-0831289 and US-Israel BSF grant 2006060.

ISSN 1433-8092

Moreover, the paper of Håstad et al. introduced concepts and techniques that now permeate the theory of pseudorandomness, such as pseudoentropy and the Leftover Hash Lemma.

A drawback of the construction of Håstad et al., however, is that it is quite complicated. While it utilizes many elegant ideas and notions, the final construction combines these in a rather ad hoc and indirect fashion due to various technical issues. In addition to being less satisfactory from an aesthetic and pedagogical perspective, the complexity of the construction also has a significant impact on its efficiency. Indeed, it is too inefficient to be implemented even for very modest settings of parameters.

In the last few years, progress has been made on simplifying the construction of Håstad et al. [16] and improving its efficiency [10]. These constructions, however, still retain the overall structure of the Håstad et al. construction, and thus retain some of the complex and ad hoc elements.

In this paper, we present a significantly more direct and efficient construction of pseudorandom generators from one-way functions. The key to our construction is a new notion of *next-block pseudoentropy*, which is inspired by the recently introduced notion of "inaccessible entropy" [13].

1.1 The HILL Construction

Informally, a function $f : \{0,1\}^n \to \{0,1\}^n$ is a *one-way function* (OWF) if it is easy to compute (in polynomial time) and hard to invert even on random inputs. A polynomial-time computable function $G : \{0,1\}^n \to \{0,1\}^{m(n)}$ is a *pseudorandom generator* (PRG) if it is stretching (i.e., $m(n) > n$) and its output distribution is pseudorandom (i.e., $G(U_n)$ is computationally indistinguishable from $U_{m(n)}$). The theorem of Håstad et al. relates these notions:

Theorem 1.1. *If there exists a one-way function, then there exists a pseudorandom generator.*

The key notion underlying their construction is the following generalization of pseudorandomness.

Definition 1.2 (pseudoentropy, informal)**.** *A random variable X has* pseudoentropy at least k *if there exists a random variable Y such that:*

1. *X is computationally indistinguishable from Y.*

2. *$\mathrm{H}(Y) \geq k$, where $\mathrm{H}(\cdot)$ denotes Shannon entropy.*[1]

A pseudoentropy generator (PEG)[2] *is a polynomial-time computable function $G : \{0,1\}^n \to \{0,1\}^{m(n)}$ such that $X = G(U_n)$ has pseudoentropy at least $\mathrm{H}(G(U_n)) + \Delta(n)$ for some $\Delta(n) \geq 1/\mathrm{poly}(n)$. We refer to $\Delta(n)$ as the* entropy gap *of G.*

That every pseudorandom generator $G : \{0,1\}^n \to \{0,1\}^{m(n)}$ is a pseudoentropy generator can be seen by taking $Y = U_{m(n)}$ and noting that $\mathrm{H}(Y) = m(n)$, but $\mathrm{H}(G(U_n)) \leq \mathrm{H}(U_n) = n$. Pseudoentropy generators are weaker in that Y may be very far from uniform, and may even have $\mathrm{H}(Y) < n$ (as long as $\mathrm{H}(G(U_n))$ is even smaller).

The construction of pseudorandom generators from one-way functions proceeds roughly in the following steps:

[1] The *Shannon entropy* of a random variable X is defined to be $\mathrm{E}_{x \overset{\mathrm{R}}{\leftarrow} X}[\log(1/\Pr[X = x])]$.

[2] Håstad et al. [14] refer to such a generator as a *false entropy generator*, and require that a pseudoentropy generator to have output pseudoentropy (at least) $n + \Delta(n)$, rather than just $\mathrm{H}(G(U_n)) + \Delta(n)$. For the informal discussion here, however, we prefer not to introduce the additional term "false entropy".

OWF to PEG: Given a one-way function $f : \{0,1\}^n \to \{0,1\}^n$, Håstad et al. define $\mathrm{PEG}(x, h, i) = (f(x), h, h(x)_{1\dots i})$, where h is an appropriate hash function and $h(x)_{1\dots i}$ denotes the first i bits of $h(x)$. PEG can be shown to be a pseudoentropy generator with an entropy gap of roughly $\Delta = \log n/n$ — Whenever $i = \log|f^{-1}(x)| + \Theta(\log n)$ (which happens with probability $\Theta(\log n/n)$) the first $\approx \log|f^{-1}(x)|$ bits of $h(x)$ extract all the entropy of x, and then we get $\Theta(\log n)$ bits of pseudoentropy by the Goldreich–Levin Hardcore-Bit Theorem [4].

Converting Shannon Entropy to Min-Entropy and Amplifying the Gap: Next, Håstad et al. use a direct product construction $\mathrm{PEG}'(x_1, \dots, x_t) = (\mathrm{PEG}(x_1), \dots, \mathrm{PEG}(x_t))$ to convert pseudoentropy into pseudo-*min*-entropy, and increase the entropy gap to be $\omega(\log n)$. This turns out to require taking roughly $t = (n/\Delta)^2$ copies.

Randomness Extraction: By hashing, Håstad et al. extract pseudorandom bits from the pseudo-min-entropy achieved so far. By also hashing the seed x to extract any remaining entropy, they obtain a pseudorandom generator. Specifically, they show that $G(x, h_1, h_2) = (h_1, h_2, h_1(\mathrm{PEG}'(x)), h_2(x))$ is a pseudorandom generator, if the output lengths of h_1 and h_2 are chosen appropriately. The choice of output lengths depends on the amount of min-entropy in the output of PEG', which in turn depends on the amount of entropy in the output of PEG. Unfortunately, these quantities may be infeasible to compute; this is handled by the next step.

Enumeration: Håstad et al. enumerate over all $u = O(n/\Delta)$ possible values k for the output entropy of PEG (up to an accuracy of $\Delta/2$), construct a pseudorandom generator G_k for each, use composition to make each G_k stretch its seed by a factor of greater than u, and then take $G(x_1, \dots, x_u) = G_1(x_1) \oplus \dots \oplus G_u(x_u)$ as their final pseudorandom generator.

The total seed length in this informal description is roughly $n \cdot t \cdot u = O(n^4/\Delta^3) = O(n^7)$. In fact, we have been cheating a bit in order to present the construction in a more modular way than in [14]. (The issues we ignored have to do with handling uniform adversaries, for which the (non-)samplability of source Y in Definition 1.2 is an issue.) The actual seed length in the main construction presented in [14] is of $O(n^{10})$ (and the construction involves additional complications). A construction of seed length $O(n^8)$ is outlined in [14], and has been formalized and proven in [16].

Above we see three main sources of inefficiency in the construction: (1) the entropy gap Δ being fairly small, (2) the conversion of Shannon entropy to min-entropy, and (3) enumerating guesses for the output entropy of the initial pseudoentropy generator. Haitner, Harnik, and Reingold [10] show how to save a factor of n in the enumeration step (by constructing a pseudoentropy generator in which more is known about how the entropy is distributed) to obtain a seed length of $O(n^7)$, but still all of the steps remain.

A further complication in the construction of Håstad et al. is that the reductions demonstrating the correctness of the construction are much more complex for uniform adversaries. This aspect of the proof has recently been simplified and made much more modular via Holenstein's uniform hardcore lemma [15, 16].

In case the one-way function is secure against exponential running time ($2^{\Omega(n)}$) adversaries, Holenstein [16] showed how to reduce the seed length to $O(n^4 \cdot \omega(\log n))$ (or $O(n^5)$ to obtain a PRG with exponential security), which was then improved by Haitner et al. [11] to $O(n \cdot \omega(\log n))$ (or $O(n^2)$ to obtain a PRG with exponential security).[3]

[3] In more detail, Holenstein's construction generalizes [14] for OWFs of "any hardness", while Haitner et al. [11]

1.2 Our Approach

Our construction is based on a generalization of the notion of a pseudoentropy generator. It is motivated by the well-known fact that the pseudorandomness of a random variable X is equivalent to each bit of X being indistinguishable from uniform given the previous ones [28]. That is, $X = (X_1, \ldots, X_n)$ is computationally indistinguishable from $U_n = (Y_1, \ldots, Y_n)$ if and only if for every i, $(X_1, X_2, \ldots, X_{i-1}, X_i)$ is computationally indistinguishable from $(X_1, X_2, \ldots, X_{i-1}, Y_i)$. It is thus natural to consider what happens if we require not that X_i be pseudorandom given the previous bits, but only that X_i has *pseudoentropy* given the previous bits. More generally, we can allow the X_i's to be blocks instead of bits.

Definition 1.3 (next-block pseudoentropy, informal). *A random variable $X = (X_1, \ldots, X_m)$ has* next-block pseudoentropy at least k *if there exists a set of random variables $Y = \{Y_1, \ldots, Y_m\}$, each jointly distributed with X, such that:*

1. *For every i, $(X_1, X_2, \ldots, X_{i-1}, X_i)$ is computationally indistinguishable from $(X_1, X_2, \ldots, X_{i-1}, Y_i)$.*
2. $\sum_i \mathrm{H}(Y_i | X_1, \ldots X_{i-1}) \geq k$.

A next-block pseudoentropy generator *(NBPEG) is a polynomial-time computable function $G : \{0,1\}^n \rightarrow (\{0,1\}^\ell)^m$ such that $(X_1, \ldots, X_m) = G(U_n)$ has next-block pseudoentropy at least $\mathrm{H}(G(U_n)) + \Delta(n)$, where again $\Delta(n)$ is called the* entropy gap.

That is, in total, the bits of X "look like" they have k bits of entropy given the previous ones. Note that the case of 1 block ($m = 1$) amounts to the definition of a pseudoentropy generator. Also note that, when $m > 1$, allowing Y to be correlated with X in this definition is essential: for example if all the blocks of X are always equal to each other (and have noticeable entropy), then there is no way to define Y that is independent of X and satisfies the first condition.

With this notion, our construction proceeds as follows.

OWF to NBPEG: Given a one-way function f, we define $G(x, h) = (f(x), h, h(x)_1, h(x)_2, \ldots, h(x)_n)$, where $h : \{0,1\}^n \rightarrow \{0,1\}^n$ is an appropriate hash function, and $h(x)_i$ is the i'th bit of $h(x)$. Notice that this is the same as the construction of Håstad et al., except that we do not randomly truncate the output. At first, this seems problematic; by revealing all of $h(x)$, it becomes easy for an adversary to compute x, and thus the pseudoentropy of output equals its real entropy (i.e., we have *zero* entropy gap). We show, however, that it does indeed have *next-block pseudoentropy at least* $n + \log n$, which is even larger than the seed length of G. We have gained in two ways here. First, the entropy gap is now $\Delta = \log n$ instead of $\Delta = \log n/n$. Second, we know the total amount of entropy in the output (though not the amount contributed by the individual blocks). These two advantages improve the complexity and security of the rest of the construction. Furthermore, the fact that the next-block pseudoentropy is larger than the seed length simplifies the construction, as we do not need to extract any additional entropy from the seed.

Entropy Equalization: Here we use a technique from [13] to convert our knowledge about the total entropy (summed over all blocks) into knowledge about the entropy in the individual

take a totally different route (based on the"randomized iterate" of a function introduced by Goldreich et al. [7]) and obtain constructions based on exponentially hard OWFs, as well as on (unknown-)regular OWFs.

blocks. We evaluate G on $u = O(n/\Delta)$ independent seeds and concatenate the outputs, but randomly shifted by $i \overset{\mathrm{R}}{\leftarrow} [n]$ coordinates. This increases our seed length and our entropy by a multiplicatives factor of approximately u, but now almost all the blocks have pseudoentropy at least the average pseudoentropy of the blocks of G.

Converting Shannon Entropy to Min-Entropy and Amplifying the Gap: This works the same as in [14]. Again we take roughly $t = O(n/\Delta)^2$ copies, but concatenate them within each block to obtain an m-block generator G'. Now each of the m blocks is indistinguishable from having high *min-entropy* conditioned on the previous ones. Thus, what we have is computational analogue of a *block source* [2], which are random sources in which each block has high min-entropy conditioned on the previous ones.

Randomness Extraction: For this step, we use a known method for block-source extraction [2, 29] and define $G(x, h) = (h, h(G'(x)_1), \ldots, h(G'(x)_m))$, where h is a universal hash function. Since we know how much pseudo-min-entropy is in each block, there is no difficulty in choosing the output length of h.

In total, our seed length is roughly $O(n \cdot u \cdot t) = O(n^4)$. For the case of exponentially hard one-way functions, we can obtain $\Delta = \Omega(n)$, and thus achieve seed $O(n \cdot \omega(\log n))$ matching [11] (but, unlike [11], our construction uses nonadaptive calls to the one-way function).

Note that our construction involves no "guessing" of entropies, neither in the construction of the initial NBPEG G, nor in an enumeration step at the end. While the entropy equalization "costs" the same (namely $u = O(n/\Delta)$) as enumeration did, it is actually doing more for us. Enumeration handled our lack of knowledge of a single entropy value (for which there were only $O(n/\Delta)$ choices), but here equalization is handling lack of knowledge for approximately n entropy values (one for each block), for which there are exponentially many choices. Moreover, enumeration required composing the pseudorandom generators to increase their stretch, resulting in a construction that is highly sequential and makes adaptive use of the one-way function. Our pseudorandom generators make nonadaptive use of the one-way function and are parallelizable (e.g., in NC^1), for getting pseudorandom generators with small stretch. Using Applebaum et al. [1], this implies the existence of pseudorandom generators in NC^0 based on the existence of one-way functions in NC^1.

1.3 Relation to Inaccessible Entropy

The notion of next-block pseudoentropy generators was inspired by the notion of *inaccessible entropy generators* in [13]. These are generators G that also produce m blocks $(x_1, \ldots, x_m)$ with the property that it is infeasible for an adversary to *generate* a sequence of blocks $(x_1, \ldots, x_m)$ that are consistent with the output of G in such a way that entropy of the individual blocks x_i is high (conditioned on the state of the adversary after generating the previous blocks). Thus, in a computational sense, the output of G has *low entropy.* For this reason, the notions of next-block pseudoentropy generators and inaccessible entropy generators seem to be dual to each other.

The initial construction of an inaccessible entropy generator in [13] is $G(x) = (f(x)_1, \ldots, f(x)_n, x)$, which is very similar to our construction of a next-block pseudoentropy generator except that there is no hashing and the bits of $f(x)$ instead of $h(x)$ are treated as separate blocks. This initial step is followed by entropy equalization and gap amplification steps that are exactly the same as the one we use (but analyzed with respect to dual notions). The final hashing step there (to construct statistically hiding commitment schemes) is more complex than ours and is necessarily interactive.

Interestingly, the notion of inaccessible entropy generator was introduced in an attempt to make the construction of statistically hiding commitment schemes from one-way functions "as simple" as the construction of pseudorandom generators from one-way functions, via manipulating notions of computational entropy. (The previous construction, from [12], was extremely complex.) In return, that effort has now inspired our simplifications and improvements to the construction of pseudorandom generators.

1.4 Paper Organization

Notations and definitions used through this paper are given in Section 2, where the new notion of a next-block pseudoentropy generator is formally defined in Section 3. In Section 4 we present our construction of next-block pseudoentropy generator from one-way functions, where in Section 5 we show how to use next-block pseudoentropy generators to get a pseudorandom generator. Finally, in Section 6 we use the above reductions to prove the main result of this paper.

2 Preliminaries

2.1 Random Variables

Let X and Y be random variables taking values in a discrete universe $\mathcal{U}$. We adopt the convention that when the same random variable appears multiple times in an expression, all occurrences refer to the same instantiation. For example, $\Pr[X = X]$ is 1. The *support* of a random variable X is $\mathrm{Supp}(X) := \{x : \Pr[X = x] > 0\}$. We write $\Delta(X, Y)$ to denote the *statistical difference* (a.k.a. variation distance) between X and Y, i.e.

$$\Delta(X,Y) = \max_{T \subseteq \mathcal{U}} \left| \Pr[X \in T] - \Pr[Y \in T] \right| .$$

If $\Delta(X,Y) \le \varepsilon$ (respectively, $\Delta(X,Y) > \varepsilon$), we say that X and Y are ε-*close* (resp., ε-far).

2.2 Entropy Measures

We will refer to several measures of entropy in this work. The relation and motivation of these measures is best understood by considering a notion that we will refer to as the *sample-entropy*: For a random variable X and $x \in \mathrm{Supp}(X)$, we define the sample-entropy of x with respect to X to be the quantity

$$\mathrm{H}_X(x) := \log(1/\Pr[X = x]).$$

The sample-entropy measures the amount of "randomness" or "surprise" in the specific sample x, assuming that x has been generated according to X. Using this notion, we can define the *Shannon entropy* $\mathrm{H}(X)$ and *min-entropy* $\mathrm{H}_\infty(X)$ as follows:

$$\mathrm{H}(X) := \mathop{\mathrm{E}}_{x \overset{\mathrm{R}}{\leftarrow} X} [\mathrm{H}_X(x)]$$

$$\mathrm{H}_\infty(X) := \min_{x \in \mathrm{Supp}(X)} \mathrm{H}_X(x)$$

Flattening Shannon Entropy. It is well-known that the Shannon entropy of a random variable can be converted to min-entropy (up to small statistical distance) by taking independent copies of this variable.

Lemma 2.1. *1. Let X be a random variable taking values in a universe $\mathcal{U}$, let $t \in \mathbb{N}$, and let $\varepsilon > 0$. Then with probability at least $1 - \varepsilon - 2^{-\Omega(t)}$ over $x \overset{R}{\leftarrow} X^t$,*

$$|\mathrm{H}_{X^t}(x) - t \cdot \mathrm{H}(X)| \leq O(\sqrt{t \cdot \log(1/\varepsilon)} \cdot \log(|\mathcal{U}| \cdot t)).$$

2. Let X, Y be jointly distributed random variables where X takes values in a universe $\mathcal{U}$, let $t \in \mathbb{N}$, and let $\varepsilon > 0$. Then with probability at least $1 - \varepsilon - 2^{-\Omega(t)}$ over $(x, y) \overset{R}{\leftarrow} (X^t, Y^t) := (X, Y)^t$,

$$\left|\mathrm{H}_{X^t|Y^t}(x|y) - t \cdot \mathrm{H}(X|Y)\right| \leq O(\sqrt{t \cdot \log(1/\varepsilon)} \cdot \log(|\mathcal{U}| \cdot t)).$$

Proof. 1. For $x = (x_1, \ldots, x_t)$, we have $\mathrm{H}_{X^t}(x) = \sum_{i=1}^t \mathrm{H}_X(x_i)$. Thus, when $x \overset{\mathrm{R}}{\leftarrow} X^t$, $\mathrm{H}_{X^t}(x)$ is the sum of t independent random variables $\mathrm{H}_X(x_i)$, and thus we can obtain concentration around the expectation (which is $t \cdot \mathrm{H}(X)$) via Chernoff-Hoeffding Bounds. These random variables $\mathrm{H}_X(x_i)$ are not bounded (as is required to apply the standard Chernoff–Hoeffding Bound), but they are unlikely to be much larger than $O(\log |\mathcal{U}|)$. Specifically, for every $\tau > 0$ we have

$$\begin{aligned}\Pr_{x_i \overset{\mathrm{R}}{\leftarrow} X}[\mathrm{H}_X(x_i) \geq \log(|\mathcal{U}|/\tau)] &\leq \sum_{x_i \in \mathcal{U}: \mathrm{H}_X(x_i) \geq \log(|\mathcal{U}|/\tau)} \Pr[X = x_i] \\ &\leq |\mathcal{U}| \cdot 2^{-\log(|\mathcal{U}|/\tau)} \\ &= \tau.\end{aligned}$$

A Chernoff Bound for random variables with such exponentially vanishing tails follows from [26], and it says that the probability that the sum deviates from the expectation by at least $\Delta \cdot (\log(|\mathcal{U}|/\tau)) + 2\tau t$ is at most $\exp(-\Omega(\Delta^2/t)) + \exp(-\Omega(\tau t))$, provided $\tau \in [0, 1]$. An appropriate choice of $\Delta = O(\sqrt{t \log(1/\varepsilon)})$ and $\tau = \min\{1, O(\log(1/\varepsilon)/t)\}$ completes the proof.

2. Similar, noting that $\mathrm{H}_{X^t|Y^t}(x|y) = \sum_{i=1}^t \mathrm{H}_{X|Y}(x_i|y_i)$.

□

2.3 One-way Functions

Definition 2.2. *Let $f\colon \{0,1\}^n \mapsto \{0,1\}^m$ be a polynomial-time computable function, where n is a security parameter and $m = m(n)$. For $T = T(n)$ and $\varepsilon = \varepsilon(n)$, we say that f is a (T, ε)-*one-way function* if for every probabilistic algorithm A running in time T and all sufficiently large n, we have:*

$$\Pr[A(Y) \in f^{-1}(Y)] \leq \varepsilon,$$

where the probability is taken over $Y = f(U_n)$ and the coin tosses of A. We say that f is a one-way function *if it is a $(p(n), 1/p(n))$-one-way function for every polynomial p.*

2.4 Pseudorandom Generators

Definition 2.3. *Let X be a random variable, depending on a security parameter n, and taking values in $\{0,1\}^m$, for $m = m(n)$. For $T = T(n)$ and $\varepsilon = \varepsilon(n)$, we say that X is (T, ε)-*pseudorandom *if for every probabilistic distinguisher D running in time T and all sufficiently large n, we have:*

$$|\Pr[D(X) = 1] - \Pr[D(U_m) = 1]| \leq \varepsilon.$$

*A polynomial-time computable function $G\colon \{0,1\}^n \mapsto \{0,1\}^m$ with $m = m(n) > n$ is a (T, ε)-*pseudorandom generator *if $G(U_n)$ is (T, ε)-pseudorandom.*

We say that X is pseudorandom *if it is $(p(n), 1/p(n))$-pseudorandom for every polynomial p. Similarly, G is a* pseudorandom generator *G if $G(U_n)$ is pseudorandom.*

3 Next-block Pseudoentropy

In this section we formally define the new notion of next-block pseudoentropy, for the cases of both Shannon entropy and min-entropy. The definitions will differ from the informal definition given in the introduction (Definition 1.3) in the following two ways, both of which are important for the treatment of uniform adversaries:

- We will require indistinguishability even against algorithms that have an oracle for sampling from the joint distribution (X, Y_i). (This enables us to show, using a hybrid argument, that pseudoentropy increases when we taking many independent copies of X. In the case of nonuniform adversaries, no oracle for sampling from (X, Y_i) is needed, as the samples can be nonuniformly hardwired into the adversary.)
- In order to achieve the first item, we will allow the random variables Y_i to depend on the distinguisher.

Similar issues arise for treating uniform adversaries with standard pseudoentropy.

Definition 3.1. *(next-block (Shannon) pseudoentropy) Let X be a random variable taking values in $\mathcal{U}^m$, where X, $\mathcal{U}$, and m may all depend on a security parameter n. For $T = T(n)$, $k = k(n)$ and $\varepsilon = \varepsilon(n)$, we say that X has (T, ε)* next-block pseudoentropy at least k *if for every oracle-aided distinguisher $D^{(\cdot)}$ of running time at most T, there exists a set of random variables $\{Y_1, \ldots, Y_m\}$ over $\mathcal{U}$ such that:*

1. *$\sum_{i=1}^m \mathrm{H}(Y_i \mid X_1, \ldots, X_{i-1}) \geq k$, and*
2. *$\mathrm{E}_{i \xleftarrow{R} [m]} \left[\Pr[D^{O_{X,Y}}(X_1, \ldots, X_i) = 1] - \Pr[D^{O_{X,Y}}(X_1, \ldots, X_{i-1}, Y_i) = 1]\right] \leq L \cdot \varepsilon$, where $O_{X,Y}(i)$, for $i \in [m]$, samples according to the joint distribution (X, Y_i), and $L = L(n)$ is a bound on number of calls made by D to $O_{X,Y}$ (including the challenge itself).*

We say that X has next-block pseudoentropy at least k, *if it has $(p(n), 1/p(n))$-next-block pseudoentropy at least k for every polynomial p. We say that* every block *of X has (T, ε)-next-block pseudoentropy at least $\alpha = \alpha(n)$, if condition (1) above is replaced with $\mathrm{H}(Y_i \mid X_{1,\ldots,i-1}) \geq \alpha$ for every $i \in [m]$.*

Note in comparing to the informal description given in the introduction, here we have omitted absolute values in the indistinguishability condition (Condition 2) above (and below). This is purely for technical convenience, as D can use $O((m/\varepsilon)^2)$ random samples from its oracle to test whether the (signed) advantages inside the expectation are positive or negative to within an accuracy of $\pm\varepsilon/2m$ and negate itself for some values of i in order to ensure a positive advantage of at least $L\varepsilon/2$.

Definition 3.2. *(next-block pseudo-min-entropy) Let X be a random variable taking values in $\mathcal{U}^m$, where X, $\mathcal{U}$, and m may all depend on a security parameter n. For $T = T(n)$, $\alpha = \alpha(n)$, and $\varepsilon = \varepsilon(n)$, we say that* every block *of X has* (T, ε)-next-block pseudo-*min*-entropy α*, if for every oracle-aided distinguisher $D^{(\cdot)}$ running in time at most $T(n)$, there exists a set of random variables $\{Y_1, \ldots, Y_m\}$ over $\mathcal{U}$ such that:*

1. $\mathrm{H}_\infty(Y_i \mid X_{1,\ldots,i-1}) \geq \alpha$*, and*
2. $\mathrm{E}_{i \xleftarrow{R} [m]} \left[\Pr[D^{O_{X,Y}}(X_1, \ldots, X_i) = 1] - \Pr[D^{O_{X,Y}}(X_1, \ldots, X_{i-1}, Y_i) = 1]\right] \leq L \cdot \varepsilon$*, where $O_{X,Y}$ and L is as in Definition 3.1.*

We say that every block *of X has* next-block pseudo-min-entropy α*, if every block of X has $(p(n), 1/p(n))$-next-block pseudo-min-entropy α, for every polynomial p.*

Unless explicitly stated otherwise, in the following sections we view a distribution over $\{0,1\}^t$ as a t-block distribution. When we refer to the next-block pseudoentropy properties of a function $G : \{0,1\}^n \rightarrow \{0,1\}^m$, these refer to the random variable $G(U_n)$.

4 One-way Functions to Next-block Pseudoentropy Generator

This section will show how to construct a next-block pseudoentropy generator ${G_{nb}}^f$ out of a one-way function $f \colon \{0,1\}^n \mapsto \{0,1\}^n$.

Theorem 4.1. *(Next-block pseudoentropy generator from one-way functions) Let n be a security parameter and $f : \{0,1\}^n \mapsto \{0,1\}^n$ be a polynomial-time computable function. Then there exists a polynomial-time computable generator $G_{nb} \colon \{0,1\}^d \mapsto \{0,1\}^m$, with $d = d(n) \in O(n)$ and $m = m(n) \in O(n)$, such that the following holds:*

Security: *Assume that f is a (T, ε) one-way function for some $T = T(n)$ and $\varepsilon = \varepsilon(n)$. Then for any* $\mathrm{poly}(n)$*-time computable value of $\varepsilon' = \varepsilon'(n) > 2^{-n/4}$, G_{nb} has $(T \cdot (\varepsilon'/n)^{O(1)}, \varepsilon')$-next-block-pseudoentropy $k = d + \log(1/\varepsilon) - O(\log n)$.*

Moreover, the reduction from the security of G_{nb} to that of f is fully black-box,[4]

Complexity: *G_{nb} is computable in* NC^1 *with one (uniformly random*[5]*) oracle call to f.*

[4] That is, the proof of security treats both f and a possible adversary as black boxes (i.e., as oracles). See [24] for more details.

[5] Here and similarly throughout the paper, we mean that when G_{nb} is evaluated on a uniformly random seed, its oracle call to f is uniformly random.

When f is a standard one-way function, we can take $T = 1/\varepsilon = n^{c'}$ for an arbitrarily large constant c' and set $\varepsilon' = 1/n^{\gamma c'}$ for a small universal constant γ, to deduce that G_{nb} has $(n^{\Omega(c')}, 1/n^{\Omega(c')})$-next-block pseudoentropy at least $k = d + c' \log n - O(\log n)$. In particular, G_{nb} has next-block pseudoentropy at least $k = d + \log n$.

Our construction employs a family of hash functions $\mathcal{Q} = \{\mathcal{Q}_n = \{q \colon \{0,1\}^n \mapsto \{0,1\}^n\}\}$. We will shortly discuss the properties needed from $\mathcal{Q}$. Given an appropriate family $\mathcal{Q}$, we can define ${G_{nb}}^f$ quite easily:

Construction 4.2. *On security parameter n, define the algorithm G_{nb} on domain $\{0,1\}^n \times \mathcal{Q}_n$, for $\mathcal{Q}_n = \{q \colon \{0,1\}^n \mapsto \{0,1\}^n\}$, and oracle $f : \{0,1\}^n \to \{0,1\}^n$ as follows:*

$$ {G_{nb}}^f(s, q) := (f(s), q, q(s)_1, \ldots, q(s)_n), $$

where s is n bits long and $q(s)_i$ denotes the ith bit of $q(s)$. (Note that we abuse notation and write q for both the function and its description.)

The following properties regarding the efficiency of ${G_{nb}}^f$ are immediate:

Lemma 4.3. *If $\mathcal{Q}$ is in NC^1 then ${G_{nb}}^f$ is in NC^1 with one oracle call to f. If the description length of $\mathcal{Q}_n$ is $O(n)$ then the input length of ${G_{nb}}^{(\cdot)}$ is linear in its first argument (as $|q| = O(|s|)$). Finally, ${G_{nb}}^f$ invokes f exactly once (and thus is non-adaptive with respect to f). On a uniformly random seed for G_{nb}, the single invocation of f is on a uniformly random input as well.*

Indeed, we will define $\mathcal{Q}$ that is both efficient and has a short description. The main requirement from $\mathcal{Q}$, however, has to do with ensuring that ${G_{nb}}^f$ is a next-block pseudoentropy generator. Let us start by presenting the following strategy showing that the entropy gap (i.e., $k - d$) is at least $\log n$ when f is a standard one-way function. Let $D_f(y) := \log |\{x \in \{0,1\}^n \colon f(x) = y\}|$ and let S be uniformly distributed over $\{0,1\}^n$. Then the distribution of S conditioned on $y = f(S)$ still has $D_f(y)$ bits of entropy. We would like $\mathcal{Q}$ to extract these bits and in addition to extract $\log n$ bits of pseudoentropy. More concretely, we ask that the first $D_f(y) + \log n$ bits of $q(S)$ are pseudorandom even given $y = f(S)$ and q (to simplify notation, we will ignore round-off errors and treat $D_f(y)$ as an integer). Given such a $\mathcal{Q}$, we are essentially done (at least when considering non-uniform security[6]). Consider the distributions $X = {G_{nb}}^f(S, Q)$ and the distribution $(Y_1, \ldots, Y_n) := (f(S), Q, R_1, \ldots R_k, Q(S)_{K+1}, \ldots, Q(S)_n)$, where S and Q are uniformly distributed, $K := D_f(f(S)) + \log n$ and the R_i's are uniformly random bits. By the above discussion, X and Y (more formally, X and $\{Y_i\}$) are next-block indistinguishable. In addition, we have:

$$
\begin{aligned}
\sum_{i=1}^{m} \mathrm{H}(Y_i \mid X_{1,\ldots,i-1}) \quad &\geq \quad \mathrm{H}(f(S)) + \mathrm{H}(Q) + \mathrm{H}(R_1, \ldots, R_K | f(S)) \\
&= \quad \mathrm{H}(f(S)) + \mathrm{H}(Q) + \mathrm{E}[D_f(f(S)) + \log n] \\
&= \quad n + \log |\mathcal{Q}_n| + \log n.
\end{aligned}
$$

and therefore ${G_{nb}}^f$ is indeed a next-block pseudoentropy generator.

The first remaining challenge is to construct such a family $\mathcal{Q}$. As we will discuss shortly, it is easy to obtain all the above properties with hash functions that have description length n^2. For

[6] I.e., the distinguisher is non-uniform and does not get oracle access to $O_{X,Y}$.

better efficiency, we will settle on $\mathcal{Q}$ with slightly weaker properties (where the pseudorandom bits extracted by $q \in \mathcal{Q}$ will be pseudorandom up to advantage $1/n$ rather than an arbitrary inverse polynomial advantage). An additional challenge is achieving next-block pseudoentropy in the (more standard) uniform setting. The difficulty is that we need X and Y to be next-block indistinguishable even given oracles that sample these distributions. While X is efficiently samplable (and thus an oracle that samples X is easy to implement), Y may not be (as $D_f(y)$ may be hard to compute). To overcome this difficulty we employ Holenstein's Uniform Hardcore Lemma [15]. Employing the Hardcore Lemma also closes the gap between the properties of $\mathcal{Q}$ we obtain and the stronger properties in the discussion.

4.1 The family $\mathcal{Q}$ and unpredictability

A family $\mathcal{Q}$ with the above properties, but with description length n^2, is easy to come by. Simply define $q(s)$ to be As, where A is a uniformly chosen $n \times n$ matrix over GF(2). For a random $y = f(S)$, the Leftover Hash Lemma [20] yields that the first $D_f(y) - c \log n$ bits of $Q(S)$ are statistically close to uniform up to statistical distance $1/n^{\Omega(c)}$. An additional $(c+1) \cdot \log n$ bits are pseudorandom by reduction to the Goldreich-Levin hardcore predicate [4]. An interesting open problem is to come up with a family $\mathcal{Q}$ that has similar properties and in addition has description length n. Instead, in this paper we relax the requirements from $\mathcal{Q}$.

Defining $q(s) = As$ is equivalent to selecting each one of the output bits of $q(s)$ to be a uniformly selected location of the Hadamard encoding of s. If instead we let each bit be a location in a polynomially-long encoding of s, we get description length $n \log n$. As long as this encoding possesses good list-decoding properties, such a construction still suffices for our purposes. To save the final $\log n$ factor, we will look at an encoding of x into logarithmically long symbols (and thus will only need $n/\log n$ symbols as the output of $q(s)$). The following lemma formalizes the properties of the encoding we will use. As with the Hadamard Code, the code we will use will satisfy both the role of extracting $D_f(y) - O(\log n)$ truly random bits via the Leftover Hash Lemma and the role of extracting $O(\log n)$ pseudorandom bits similarly to a hardcore function.

Lemma 4.4. *There exists an* NC^1 *algorithm* Enc *such that on input* $s \in \{0,1\}^n$*, the algorithm* Enc *produces* $t = \mathrm{poly}(n)$ *symbols* $\mathrm{Enc}(s)_1, \mathrm{Enc}(s)_2, \ldots, \mathrm{Enc}(s)_t$ *with each* $\mathrm{Enc}(s)_i \in \{0,1\}^\ell$ *for* $\ell = \lceil \log n \rceil$ *and such that the following properties hold:*

Almost 2-Universal: *For every two distinct n-bit strings $s \neq s'$ it holds that*

$$\Pr_{i \in [t]}[\mathrm{Enc}(s)_i = \mathrm{Enc}(s')_i] \leq 2^{-\ell} \cdot (1 + 1/(2n^5)).$$

List Decoding: *There exists a polynomial-time algorithm* Dec *that on input* 1^n *and given oracle access to a function* $\tilde{A} : [t] \times \{0,1\}^\ell \to \{0,1\}$*, outputs a list of* $\mathrm{poly}(n)$ *strings that includes every* $s \in \{0,1\}^n$ *satisfying the following:*

$$\Pr_{i \xleftarrow{R} [t]}[\tilde{A}(i, \mathrm{Enc}(s)_i) = 1] - \Pr_{i \xleftarrow{R} [t], z \xleftarrow{R} \{0,1\}^\ell}[\tilde{A}(i, z) = 1]] > 1/5n^2.$$

Note that the oracle $\tilde{A}$ has a domain of size $t \cdot 2^\ell = \mathrm{poly}(n)$, so Dec has enough time to query it on all inputs (i.e. "local decoding" is not needed).

Proof Sketch. Enc can be taken to be the concatenation of the Reed-Solomon Code of degree $d = n-1$ over $\mathbb{F} = \mathrm{GF}(2^{c\ell})$ with the Hadamard Code over $\mathbb{F}' = \mathrm{GF}(2^{\ell})$, where $\ell = \lceil \log n \rceil$ and $c \in \mathbb{N}$ is a sufficiently large constant to be determined below. That is, codewords are of length $t = |\mathbb{F}|^2 = \mathrm{poly}(n)$, and for $s = (s_0, \ldots, s_{n-1}) \in \{0,1\}^n$ and $(a,b) \in \mathbb{F}^2$, we set $\mathrm{Enc}(s)_{a,b} = (\sum_{i=0} s_i a^i) \odot b \in \mathbb{F}'$, where $\odot$ denotes dot product (viewing elements of $\mathbb{F}$ as c-dimensional vectors over $\mathbb{F}'$).

The almost-universality property follows from a standard argument: the probability that $\mathrm{Enc}(s)_{a,b} = \mathrm{Enc}(s')_{a,b}$ is bounded by the probability that a random symbol of two distinct Reed-Solomon codewords agree (which is at most $d/|\mathbb{F}| < n/n^c$) plus the probability that a random symbol in two distinct Hadamard codewords agree (which is $1/|\mathbb{F}'| = 1/2^{\ell}$). By taking c sufficiently large (e.g. $c \geq 8$), we get the desired property.

For the list-decoding property, given $\tilde{A} : [t] \times \{0,1\}^{\ell} \to \{0,1\}$ that can distinguish a random symbol of some (unknown) codeword $\mathrm{Enc}(s)$ from uniform with advantage greater than $1/5n^2$, we will show how to obtain from $\tilde{A}$ a list $k = \mathrm{poly}(n)$ of "received words" $r_1, \ldots, r_k \in (\{0,1\}^{\ell})^t$ such $\mathrm{Enc}(s)$ agrees with one of the r_i's in more than a $\alpha = (1 + 1/5n^2)/2^{\ell}$ fraction of positions. By applying the list-decoding algorithm of [8, Thm. 7] to each of the r_i's, we are guaranteed to recover each such s provided that

$$\alpha \geq \frac{1}{|\mathbb{F}'|} + \sqrt{\frac{d}{|\mathbb{F}|}} + O\left(\frac{1}{t}\right).$$

Since $\alpha = 1/|\mathbb{F}'| + \Omega(1/n^3)$, and $\sqrt{d/|\mathbb{F}|} + O(1/t) = O(\sqrt{1/n^{c-1}})$, the inequality is satisfied for a sufficiently large choice of the constant c (e.g. $c = 11$).

We still need to show how to obtain the received words $r_1, \ldots, r_k$ from $\tilde{A}$. For this, we use the standard transformation from distinguishers to predictors [28]. For each pair $u, v \in \mathbb{F}'$, we define

$$r_{u,v}[i] = \begin{cases} u & \text{if } \tilde{A}(i,u) = 1 \\ v & \text{otherwise.} \end{cases}$$

Now consider any s such that

$$\Pr_{i \xleftarrow{\mathrm{R}} [t]}[\tilde{A}(i, \mathrm{Enc}(s)_i) = 1] - \Pr_{i \xleftarrow{\mathrm{R}} [t], z \xleftarrow{\mathrm{R}} \{0,1\}^{\ell}}[\tilde{A}(i,z) = 1]] > 1/5n^2.$$

Let's analyze the expected agreement between s and $r_{u,v}$, when u and v are uniformly random distinct elements of $\{0,1\}^{\ell}$.

$$\begin{aligned}
\Pr_{u,v,i}[r_{u,v}[i] = \mathrm{Enc}(s)_i] &= \Pr_{u,v,i}[u = \mathrm{Enc}(s)_i \wedge \tilde{A}(i, \mathrm{Enc}(s)_i) = 1] + \Pr_{u,v,i}[v = \mathrm{Enc}(s)_i \wedge \tilde{A}(i,u) = 0] \\
&= \frac{1}{2^{\ell}} \cdot \Pr_i[\tilde{A}(i, \mathrm{Enc}(s)_i) = 1] + \frac{1}{2^{\ell}} \cdot \Pr_{i, u \neq \mathrm{Enc}(s)_i}[\tilde{A}(i,u) = 0] \\
&= \frac{1}{2^{\ell}} \cdot \left(\Pr_i[\tilde{A}(i, \mathrm{Enc}(s)_i) = 1] + 1 - \Pr_{i, u \neq \mathrm{Enc}(s)_i}[\tilde{A}(i,u) = 1] \right) \\
&> \frac{1}{2^{\ell}} \cdot \left(1 + \Pr_i[\tilde{A}(i, \mathrm{Enc}(s)_i) = 1] - \Pr_{i,u}[\tilde{A}(i,u) = 1] \right) \\
&> \frac{1 + 1/5n^2}{2^{\ell}}
\end{aligned}$$

□

Construction 4.5. *Let n, Enc, t and ℓ be as in Lemma 4.4. The description of a random hash function $q \in \mathcal{Q}_n$ is composed of $\lceil n/\ell \rceil$ random indices $i_1, \ldots, i_{\lceil n/\ell \rceil} \in [t]$. On input s define $q(s) = \mathrm{Enc}(s)_{i_1} \ldots \mathrm{Enc}(s)_{i_{\lceil n/\ell \rceil}}$ (which for simplicity will be assumed to be exactly n-bit long).*

Lemma 4.6. *Let n be a security parameter, $\mathcal{Q}$ be as in Construction 4.5, let G_{nb} be the oracle-aided algorithm for Construction 4.2 (with respect to $\mathcal{Q}$), and let $f : \{0,1\}^n \to \{0,1\}^n$ be a (T, ε) one-way function, for $T = T(n) \geq n, \varepsilon = \varepsilon(n)$. Then there exists a constant $c > 0$ such that*

$$\sum_{i=0}^{n-1} \Pr[P\left(f(S), Q, Q(S)_1, \ldots, Q(S)_i\right) \neq Q(S)_{i+1}] \geq \frac{n - \mathrm{H}(f(S)) + \log(1/\varepsilon) - c \log n}{2},$$

even when P is allowed to run in time T/n^c, where S and Q are uniformly distributed over $\{0,1\}^n$ and $\mathcal{Q}_n$ respectively.

Note that above (and below) $Q(S)_1, \ldots, Q(S)_{i+1}$ refer to individual *bits* of $Q(S)$, not ℓ-bit blocks.

Proof. Consider the following two distributions:

- $X := (f(S), Q, Q(S)_1, \ldots Q(S)_K)$ where K is the random variable $D_f(f(S)) + \log(1/\varepsilon) - c' \log n$, and
- $Y := (f(S), Q, R_1, \ldots, R_K)$, where $R_1, \ldots, R_K$ are uniformly random and independent bits.

We will show that, for a sufficiently large choice of the constant c', no algorithm with running time $T' = T/n^{c'}$ can distinguish X from Y with advantage better than $1/n$. The lemma then follows by observing that the sum of the error probabilities in predicting the bits $R_1, \ldots, R_K$ is at least $\mathrm{E}[K]/2$. In more detail, first note that K is always smaller than n, because if $D_f(t) \geq n - \log(1/\varepsilon)$ for some t, then $\Pr[U_n \in f^{-1}(t)] \geq \varepsilon$, which gives rise to a trivial inversion algorithm that succeeds with probability at least ε. Thus,

$$\begin{aligned}
&\sum_{i=0}^{n-1} \Pr[P\left(f(S), Q, Q(S)_1, \ldots, Q(S)_i\right) \neq Q(S)_{i+1}] \\
&\quad \geq \sum_{i=0}^{n-1} \Pr[i < K \wedge (P\left(f(S), Q, Q(S)_1, \ldots, Q(S)_i\right) \neq Q(S)_{i+1})] \\
&\quad \geq \sum_{i=0}^{n-1} (\Pr[i < K \wedge (P\left(f(S), Q, R_1, \ldots, R_i\right) \neq R_{i+1})] - 1/n) \\
&\quad \geq \sum_{i=0}^{n-1} (\Pr[i < K]/2 - 1/n) \\
&\quad = \mathrm{E}[K]/2 - 1 \\
&\quad = (n - \mathrm{H}(f(S)) + \log(1/\varepsilon) - c' \log n - 2)/2.
\end{aligned}$$

We will assume for simplicity that ℓ always divides both $D_f(t)$ and $\log(1/\varepsilon)$. Let $A = D_f(f(S))/\ell$ and $a' = \log(1/\varepsilon)/\ell$. Recall that $q(s)$ is defined to be $(\mathrm{Enc}(s)_{i_1} \ldots \mathrm{Enc}(s)_{i_{\lceil n/\ell \rceil}})$. Therefore, we need to prove that $(\mathrm{Enc}(S)_{I_1} \ldots \mathrm{Enc}(S)_{I_{A+a'-c'}})$ is $1/n$-pseudorandom even given $f(S)$ and $Q = (I_1, \ldots, I_{\lceil n/\ell \rceil})$. To prove this, we introduce a hybrid distribution Z, defined as follows:

- $Z := (f(S), Q, \mathrm{Enc}(S)_{I_1}, \ldots, \mathrm{Enc}(S)_{I_{A-5}} Z_{A-4} \cdots Z_{A+a'-c'})$, where $Z_1, \ldots, Z_{A+a'-c'}$ are uniformly random and independent elements of $\{0,1\}^{\ell}$.

First we argue that Z has statistical distance at most $1/10n^2$ from $Y = (f(S), Q, Z_1, \ldots, Z_{A+a'-c'})$. Let's condition on any fixed value $f(S) = t$. This also determines that $A = D_f(t)/\ell =: a$. The distribution of S conditioned on $f(S) = t$ still has min-entropy $D_f(t) = a \cdot \ell$. As Enc is almost universal (see first property in Lemma 4.4), for every two distinct n-bit strings $s \neq s'$ we have that

$$\Pr[\mathrm{Enc}(s)_{I_1} \ldots \mathrm{Enc}(s)_{I_{a-5}} = \mathrm{Enc}(s')_{I_1} \ldots \mathrm{Enc}(s')_{I_{a-5}}] \leq 2^{-(a-5)\ell} \cdot (1 + 1/100n^4).$$

Therefore, the Leftover Hash Lemma [20, 19] yields that (conditioned on $f(S) = t$) the prefix $(Q, \mathrm{Enc}(S)_{I_1} \ldots \mathrm{Enc}(S)_{I_{A-5}})$ is δ-close to uniform for $\delta = 1/2 \cdot \sqrt{2^{-5\ell} + 1/100n^4} \leq 1/10n^2$.

Now we argue that there is no algorithm B that runs in time T' and distinguishes $X = (f(S), Q, \mathrm{Enc}(S)_{I_1}, \ldots, \mathrm{Enc}(S)_{I_{A+a'-c'}})$ from Z with advantage better than $9/10n$. Assume that there is such an algorithm A. By an hybrid argument, we get that for a random index J between $A-4$ and $A+a'-c'$ the algorithm B distinguishes with advantage at least $9/10n^2$ the distributions H^J and H^{J+1} where

- $H^j = (f(S), Q, \mathrm{Enc}(S)_{I_1}, \ldots, \mathrm{Enc}(S)_{I_{j-1}}, Z_j \cdots Z_{A+a'-c'})$.

Now, by a Markov argument, it holds that with probability at least $1/2n^2$ over fixing B's coin tosses and all components of H^J and H^{J+1} other than I_J, $\mathrm{Enc}(S)_{I_J}$, and Z_J, B distinguishes $(I_J, \mathrm{Enc}(S)_{I_J})$ from (I_J, Z_J) with advantage at least $2/5n^2$. Thus, if we can generate those components given a value of $f(S)$, we can plug B into the decoder Dec guaranteed by Lemma 4.4 to recover S and hence invert f. We will not generate the components exactly, but closely enough so that the inversion will still succeed with nonnegligible probability.

The inverter gets as input $t = f(S)$ and operates as follows:

1. Guess the value of $a \overset{\mathrm{R}}{\leftarrow} [n/\ell]$.
2. Sample uniformly at random coin tosses for B.
3. Sample uniformly at random $i_1 \ldots i_{a-5} \overset{\mathrm{R}}{\leftarrow} [t])$ and $z_1, \ldots z_{i_{a-5}} \overset{\mathrm{R}}{\leftarrow} \{0,1\}^{\ell}$.
4. Select at random $j \overset{\mathrm{R}}{\leftarrow} \{a-4, \ldots, a+a'-c'\}$.
5. Sample uniformly at random $i_{a-4}, \ldots i_{j-1}, i_{j+1}, \ldots i_{a+a'-c'} \overset{\mathrm{R}}{\leftarrow} [t]$ and $z_{a-4}, \ldots z_{j-1}, z_{j+1}, \ldots z_{a+a'-c'} \overset{\mathrm{R}}{\leftarrow} \{0,1\}^{\ell}$.
6. We have fixed all of the inputs of B aside from i_j and z_{i_j}. Denote the resulting algorithm after all of these fixings as $\tilde{B}$.
7. Invoke $\mathrm{Dec}^{\tilde{B}}(1^n)$, where Dec is as in Lemma 4.4.
8. Select at random an element in the list returned by $\mathrm{Dec}^{\tilde{B}}$.

Let's examine how the distribution of B's inputs that are fixed by the inverter differs from that in H^J and H^{J+1}:

- The inverter chooses a uniformly at random. This value will be correct (i.e. equal to $\ell \cdot D_f(t)$) with probability at least $1/n$.
- The inverter chooses $z_1, \ldots z_{i_{a-5}}$ uniformly at random, whereas in H^J these are distributed according to $(\mathrm{Enc}(S)_{I_1}, \ldots, \mathrm{Enc}(S)_{I_{a-5}})$. However, by the indistinguishability of Z and Y, the latter distribution is $(1/10n^2)$-close to uniform (given $f(S)$ and Q). Thus, this will reduce the probability of a good fixing by at most $1/10n^2$, from $1/2n^2$ to $2/5n^2$.
- The inverter chooses $z_{a-4}, \ldots z_{j-1}$ uniformly at random whereas in H^J these are distributed according to $(\mathrm{Enc}(S)_{I_{A-4}}, \ldots, \mathrm{Enc}(S)_{I_{J-1}})$. However, this amounts to guessing at most $u = (a' + 5 - c') \cdot \ell$ bits, and hence will be correct with probability at least $2^{-u} \geq \varepsilon \cdot n^{c'}/n^5$.

In total, the list-decoding algorithm successfully recovers S (among a list of $\mathrm{poly}(n)$ possibilities) with probability at least $(1/n) \cdot (2/5n^2) \cdot (\varepsilon \cdot n^{c'}/n^5) = \varepsilon \cdot n^{c'} / \mathrm{poly}(n)$. Choosing at random from the list of decodings reduces the inversion probability by another factor of $\mathrm{poly}(n)$. Taking c' sufficiently large, the inverter succeeds with probability at least ε. We note that the inverter runs in time $\mathrm{poly}(n)$ making at most $2^\ell \cdot t = \mathrm{poly}(n)$ queries to B. Thus if B runs in time $T(n)/n^c$ for a sufficiently large constant c, the inverter runs in time at most $T(n)$, contradicting the security of f. □

4.2 Proving Next-block Pseudoentropy via Hardcore Lemma

Lemma 4.6 shows that the output of G_{nb} (after $f(S)$ and Q) satisfies a weak next-bit unpredictability. In this section, we use Holenstein's uniform variant of Impagliazzo's Hardcore Lemma [18] to translate this weak next-bit unpredictability into next-bit pseudoentropy, thereby proving Theorem 4.9.

The Hardcore Lemma translates weak unpredictability (of a single bit) into strong unpredictability on a noticeable fraction of inputs:

Proposition 4.7 ([17]). *Let n be a security parameter, and let $h\colon \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}$ and $V\colon \{0,1\}^n \mapsto \{0,1\}$ be polynomial-time computable functions. Let $\delta_0 = \delta_0(n) \in [0,1]$, $\delta = \delta(n) \in [\delta_0, 1]$, and $\gamma = \gamma(n) \in [0,1] > 2^{-n/3}$. Assume that*

$$\Pr[M(\delta_0, \gamma, h(U_n)) = V(U_n)] \leq 1 - \delta/2$$

for every probabilistic algorithm M running in time $T = T(n)$ and large enough n. Then for every oracle-aided predictor P running in time $T \cdot (\gamma\delta_0/n)^{O(1)}$ and all sufficiently large n, there exists a set $L \subseteq \{0,1\}^n$ of density at least δ such that

$$\Pr_{W \overset{R}{\leftarrow} L}[P^{\chi_L}(h(W)) = V(W)] < (1+\gamma)/2,$$

where χ_L is the characteristic function of L, provided that all the queries of P to χ_L are computed independently of the input $h(W)$. Furthermore, the reduction is fully black-box.

The above differs from Holenstein's Hardcore Lemma in that it does not require that δ be known to M, but only requires a lower bound δ_0 on δ. However, the above can be readily deduced from Holenstein's version; see Appendix A.

We now reinterpret Holenstein's Hardcore Lemma in terms of conditional pseudoentropy, similarly to the reinterpretation of Impagliazzo's Hardcore Lemma in [25].

Proposition 4.8. *Let n be a security parameter, $\delta_0 = \delta_0(n) \in [0,1]$, $\delta = \delta(n) \in [\delta_0, 1]$, and $\gamma = \gamma(n) \in [0,1] > 2^{-n/3}$. Let (A, B) be a* $\mathrm{poly}(n)$*-time samplable random variable over $\{0,1\}^{\ell(n)} \times \{0,1\}$ such that*

$$\Pr[M(\delta_0, \gamma, A) = B] \leq 1 - \delta/2$$

for every probabilistic algorithm M running in time $T = T(n)$ and large enough n. Then for every oracle-aided distinguisher D running in time $T' = T \cdot (\delta_0\gamma/n)^{O(1)}$ and all sufficiently large n, there is a random variable C, jointly distributed with (A, B), such that:

1. $\mathrm{H}(C|A) \geq \delta$.
2. $\Pr[D^{O_{A,B,C}}(A, B) = 1] - \Pr[D^{O_{A,B,C}}(A, C) = 1] \leq \gamma$,

where $O_{A,B,C}$ is an oracle that samples according to the joint distribution (A, B, C).

Proof. Let $(h, V) : \{0,1\}^n \to \{0,1\}^{\ell(n)} \times \{0,1\}$ be the poly-time sampling algorithms for (A, B), i.e. $(h(U_n), V(U_n)) = (A, B)$. (By renaming the security parameter n, we may assume that the sampling algorithms use n coin tosses.) Thus we may apply Proposition 4.7 to the pair (h, V). For any given subset $L \subseteq \{0,1\}^n$ of density δ, we define a probabilistic function $V_L : \{0,1\}^n \to \{0,1\}$, where

$$V_L(r) = \begin{cases} V(r) & \text{if } r \notin L \\ \text{a random bit} & \text{if } r \in L. \end{cases}$$

From this we get a random variable C_L jointly distributed with (A, B), defined by $(A, C_L) = (h(U_n), V_L(U_n))$. Notice that $\mathrm{H}(C_L|A)$ is at least the density of L, namely δ. We will show that taking $C = C_L$ for some L suffices. Suppose for contradiction that we have an oracle-aided distinguisher D running in time T' such that for every L of density δ, $\Pr[D^{O_{A,B,C_L}}(A, B) = 1] - \Pr[D^{O_{A,B,C_L}}(A, C_L) = 1] > \gamma$. Since B and C_L are identical when $U_n \notin L$, we have

$$\Pr[D^{O_{A,B,C_L}}(A, B) = 1 | U_n \in L] - \Pr[D^{O_{A,B,C_L}}(A, C_L) = 1 | U_n \in L] > \gamma.$$

Since C_L is a uniformly random bit when $U_n \in L$, we can apply the standard reduction from distinguishing to predicting to obtain an oracle-aided predictor P, running in time $T' + O(1)$ such that

$$\Pr[P^{\chi_L}(A) = B | U_n \in L] > (1 + \gamma)/2. \qquad (1)$$

Specifically, on input x, P generates a random bit $b \overset{\mathrm{R}}{\leftarrow} \{0,1\}$, runs $D^{O_{A,B,C_L}}(x, b)$, outputs b if D outputs 1, and outputs $\neg b$ if D outputs 0. P can simulate random samples from the oracle O_{A,B,C_L} by choosing $r \overset{\mathrm{R}}{\leftarrow} \{0,1\}^n$ and outputting $(h(r), V(r), V_L(r))$, which can be efficiently computed using P's oracle access to χ_L. Equation (1) can be rewritten as:

$$\Pr_{W \overset{\mathrm{R}}{\leftarrow} L}[P^{\chi_L}(h(W)) = V(W)] > (1 + \gamma)/2.$$

This contradicts Proposition 4.7. □

We now use this form of the Hardcore Lemma to deduce Theorem 4.9 from Lemma 4.6.

Theorem 4.9 (Theorem 4.9, restated)**.** *Let n be a security parameter and $f : \{0,1\}^n \mapsto \{0,1\}^n$ be a polynomial-time computable function. Then there exists a polynomial-time computable generator $G_{nb} \colon \{0,1\}^d \mapsto \{0,1\}^m$, with $d = d(n) \in O(n)$ and $m = m(n) \in O(n)$, such that the following holds:*

Security: *Assume that f is a (T, ε) one-way function for some $T = T(n)$ and $\varepsilon = \varepsilon(n)$. Then for any* $\mathrm{poly}(n)$*-time computable value of $\varepsilon' = \varepsilon'(n) > 2^{-n/4}$, G_{nb} has $(T \cdot (\varepsilon'/n)^{O(1)}, \varepsilon')$-next-block-pseudoentropy $k = d + \log(1/\varepsilon) - O(\log n)$.*

Moreover, the reduction from the security of G_{nb} to that of f is fully black-box,[7]

Complexity: *G_{nb} is computable in* NC^1 *with one (uniformly random*[8]*) oracle call to f.*

Proof. (of Theorem 4.9) Let $W = (Q, S)$ be uniformly distributed over the domain of G_{nb} (i.e., $\{0,1\}^n \times \mathcal{Q}_n$), let $X = G_{nb}(W)$ and denote the length of X by $m = m(n) = 2n + \log|\mathcal{Q}_n|$. Assume that the the theorem does not hold. In particular, there exists an oracle-aided algorithm D whose running time is running time is bounded by $T' = T'(n)$ and the following holds for every set of distributions $\{Y_i\}_{i \in [m]}$ over $\{0,1\}$, jointly distributed with X, with $\sum_{i=1}^{m} \mathrm{H}(Y_i \mid X_1, \ldots, X_{i-1}) \geq k = d + \log(1/\varepsilon) - c \cdot \log n$:

$$\mathop{\mathrm{E}}_{i \overset{\mathrm{R}}{\leftarrow} [m]} \left[\Pr[D^{O_{X,Y}}(X_1, \ldots, X_i) = 1] - \Pr[D^{O_{X,Y}}(X_1, \ldots, X_{i-1}, Y_i) = 1] \right] \geq L \cdot \varepsilon' \geq \varepsilon', \tag{2}$$

where L is the number of oracle queries made by D and c is the constant of Lemma 4.6.

Recall that $X = (f(S), Q, Q(S)_1, \ldots, Q(S)_n)$. By Lemma 4.6, if we take $A = (f(S), Q, Q(S)_1, \ldots, Q(S)_{I-1})$ and $B = Q(S)_I$, for I uniformly random over $[n]$, we have

$$\Pr[M(\varepsilon', A) = B] \leq 1 - \delta/2, \tag{3}$$

for

$$\delta = \tfrac{n - \mathrm{H}(f(S)) + \log(1/\varepsilon) - c \log n}{n}$$

and any algorithm M running in time $T/n^c - p(n) \leq T/2 \cdot n^c$, where $p \in \mathrm{poly}$ is the computing time of ε', and for the inequality we assume without loss of generality that $T > 2p(n) \cdot n^c$ (otherwise the conclusion of the theorem holds trivially). In the following we use Proposition 4.8 for showing that Equation (3) is in contradiction to Equation (2).

Let C be any random variable, jointly distributed with A, such that $\mathrm{H}(C|A) \geq \delta$. We can use any such C to define a set $\{Y_i\}$ of random variables jointly distributed with X:

$$Y_i|_{X=(x_1,\ldots,x_m)} = \begin{cases} x_i & \text{if } i \leq n + \log|\mathcal{Q}| \\ C|_{I=i, X=(x_1,\ldots,x_n)} & \text{otherwise.} \end{cases}$$

InoteCan $C|_{I=i}$ not be well defined?

Then

$$\begin{aligned} \sum_{i=1}^{m} \mathrm{H}(Y_i|X_1, \ldots, X_{i-1}) &= \sum_{i=1}^{n+\log|\mathcal{Q}|} \mathrm{H}(X_i|X_1, \ldots, X_{i-1}) + \sum_{i=1}^{n} \mathrm{H}(C|A, I = i) \\ &= \mathrm{H}(f(S), Q) + n \cdot \mathrm{H}(C|A) \\ &\geq \mathrm{H}(f(S)) + \mathrm{H}(Q) + n - \mathrm{H}(f(S)) + \log(1/\varepsilon) - c \log n \\ &= d + \log(1/\varepsilon) - c \log n = k, \end{aligned}$$

[7] That is, the proof of security treats both f and a possible adversary as black boxes (i.e., as oracles). See [24] for more details.

[8] Here and throughout the paper, when we say that G_{nb} makes a uniformly random call to f, we refer to the distribution of the call when we run G_{nb} on a uniformly random seed.

where d is the seed length of G_{nb}. Therefore Equation (2) holds, i.e. D is a next-block distinguisher for X and $\{Y_i\}$ with advantage ε'. Observe that D has zero advantage when $i \leq n + \log|\mathcal{Q}|$ (because then $Y_i = X_i$), and thus must gain its entire advantage when $i > n + \log|\mathcal{Q}|$. In the latter case, D is distinguishing (A, B) from (A, C) with advantage ε'. Moreover, each of D's oracle queries to $O_{X,Y}(i)$ can be simulated with $O(n)$ queries to $O_{A,B,C}$. (It may take an expected $O(n)$ trials to get a sample in which I has the desired value.) Thus, we obtain an algorithm D_0 running in time $T_0 = O(T' \cdot n)$ such that $D_0^{O_{A,B,C}}$ distinguishes (A, B) from (A, C) with advantage greater that ε', for any C such that $\mathrm{H}(C|A) \geq \delta$. Proposition 4.8 yields the existence of an algorithm M of running time $T_0 \cdot (n/\varepsilon')^{O(1)}$ such that $\Pr[M(\varepsilon', A) = B] > 1 - \delta/2$ for infinitely many n's. (We can set $\delta_0 = 1/n$, since if $\delta < 1/n$, the conclusion of the theorem holds trivially.) Setting the constant in the definition of T' to large enough value (recall that $T' = T \cdot (\varepsilon'/n)^{O(1)}$) yields that the running time of M is bounded by $T/2 \cdot n^c$, in contradiction to Equation (3). □

5 From Next-Block Pseudoentropy to Pseudorandom Generators

In this section we show how to transform a next-block pseudoentropy generator into a pseudorandom generator.

Theorem 5.1. *(Next-block pseudoentropy generator to pseudorandom generator) Let n be a security parameter, and let $m = m(n)$, $\Delta = \Delta(n) \in [1/\operatorname{poly}(n), n]$, and $\kappa = \kappa(n) \in \{1, \ldots, n\}$ be* poly(n)*-time computable. For every polynomial-time computable, m-block generator $G_{nb} \colon \{0,1\}^n \mapsto \{0,1\}^m$, there exists a polynomial-time computable generator $G : \{0,1\}^d \to \{0,1\}^{d \cdot (1+\Omega(\Delta/n))}$ with seed length*

$$d = d(n) = O\left(\frac{n^2 \cdot m^2 \cdot \kappa \cdot \log^2 n}{\Delta^3}\right)$$

such that the following holds:

Security: *If G_{nb} has (T, ε)-next-block pseudoentropy at least $n + \Delta$, for $T = T(n), \varepsilon = \varepsilon(n)$, then G is a $(T - n^{O(1)}, n^{O(1)} \cdot (\varepsilon + 2^{-\kappa}))$-pseudorandom generator. Moreover, the reduction is fully black-box.*

Complexity: *G is computable in NC^1 with $O(d/n)$ (uniformly random) oracle calls to G_{nb}.*

In Theorem 5.1, it may be convenient to view $\kappa(n)$ as the security parameter of the construction. In particular, when $\kappa(n)$ logarithmic in $1/\varepsilon(n)$ we get that $(T(n), \varepsilon(n))$-next-block pseudoentropy turns into an $(T(n)/\operatorname{poly}(n), \operatorname{poly}(n) \cdot \varepsilon(n))$-pseudorandom generator.

We prove Theorem 5.1 via the following sequence of reductions:

1. In Section 5.1 we show how to get a better handle on the output distribution of the G_{nb} — specifically, we apply a generic transformation on G_{nb}, to get a generator for which the (conditional) pseudoentropy of each of its output blocks is the same (i.e., $(n + \Delta)/m$).

2. In Section 5.2 we consider a direct product of the latter next-block pseudoentropy generator, and show that this action both increases the absolute gap between the next-block pseudoentropy of the generator and its real entropy (i.e., its input length), and transforms its next-block pseudoentropy into next-block pseudo-min-entropy.

3. In Section 5.3 we show to extract pseudorandomness from the output of the latter type of generators.
4. In Section 5.4, we put the above parts together to prove Theorem 5.1.

To simplify notations, we prove the first three steps with respect to arbitrary next-block pseudoentropy distributions. Given a distribution X over $\mathcal{U}^m$, a set of distributions $Y = \{Y_i\}_{i\in[m]}$ over $\mathcal{U}$, and an oracle-aided algorithm $D^{(\cdot)}$, we let $\delta^D_{X,Y} := \mathrm{E}_{i \xleftarrow{\mathrm{R}} [m(n)]}\left[\delta^D_{X,Y,i} := \Pr[D^{O_{X,Y}}(X_1,\ldots,X_i) = 1] - \Pr[D^{O_{X,Y}}(X_1,\ldots,X_{i-1},Y_i) = 1]\right]$, where $O_{X,Y}(i)$ samples according to the joint distribution (X, Y_i) (see Definition 3.1). Finally, in all of the following claims we assume the description of the "universe" $\mathcal{U}$ is polynomial in n.

5.1 Entropy Equalization

In this section we show to manipulate a given distribution to gain a better characterization of its next-block pseudoentropy, without losing "too much" pseudoentropy. The following transformation is closely related to a similar reduction from [9]. The idea is the following: consider an m-block random variable X over $\mathcal{U}^m$ with next-block pseudoentropy k. Now generate $m \cdot \ell$ blocks by concatenating ℓ independent copies one after the other. Finally, for a random $j \in [m]$, erase the first j blocks and the last $m - j$ blocks. We now have a new variable $\widetilde{X}$ with $m \cdot (\ell - 1)$ blocks and for every location i the block in the ith location of $\widetilde{X}$ is a block of X in a random location. It is not hard to prove (as we do below) that the next-block pseudoentropy of each block is at least k/m. On the other hand, the real entropy of $\widetilde{X}$ is at most ℓ times that of X. Taking large enough ℓ we get that the (relative) difference between next-block pseudoentropy and real entropy has not significantly decreased.

For $j \in [m]$ and $z^{(1)},\ldots,z^{(\ell)} \in \mathcal{U}^m$, we let $\mathrm{Eq}(j, z^{(1)},\ldots,z^{(\ell)}) := (z^{(1)}_j,\ldots,z^{(1)}_m,\ldots,z^{(\ell)}_1,\ldots,z^{(\ell)}_{j-1})$.

Lemma 5.2. *Let n be a security parameter, and let $m = m(n) = \mathrm{poly}(n)$ and $\ell = \ell(n) = \mathrm{poly}(n)$ be $\mathrm{poly}(n)$-time computable integer functions, where $\ell(n) > 1$. Let X be random variable over $\mathcal{U}^m$ with (T,ε)-next-block pseudoentropy at least k, for $T = T(n)$, $\varepsilon = \varepsilon(n)$ and $k = k(n)$. Let J be uniformly distributed over $[m]$ and let $\widetilde{X} = \mathrm{Eq}(J, X^{(1)},\ldots,X^{(\ell)})$, where the $X^{(i)}$'s are iid copies of X. Then every block of $\widetilde{X}$ has $(T - O(\ell \cdot m \cdot \log|\mathcal{U}|), \ell \cdot \varepsilon)$ next-block pseudoentropy at least k/m. Moreover, the reduction between the security of $\widetilde{X}$ and X is fully black-box.*

Proof. Let $m' = (\ell - 1)\cdot m$ and let $Y = \{Y_1,\ldots,Y_m\}$ be a set of random variable jointly distributed with X. In the following we think of Y as a single random variable $Y = (Y_1,\ldots,Y_m)$ jointly distributed with X, though we only sample a single entry Y_i per instance of Y. Let $Y^{(1)},\ldots,Y^{(\ell)}$ be iid copies of Y and let $\widetilde{Y} = \mathrm{Eq}(J, Y^{(1)},\ldots,Y^{(\ell)})$ be jointly distributed with $\widetilde{X}$ in the natural way —- J takes the same value as in $\widetilde{X}$, and for every $j \in [\ell]$, $Y^{(j)}$ is jointly distributed with $X^{(j)}$ according to the joint distribution (X, Y). Notice that $\widetilde{Y}_i = Y_{J+i-1 \bmod m}$ (where we define $m \bmod m$ to equal m rather than 0), and that $J + i - 1$ is uniformly distributed in $[m]$.

Thus, for every $i \in [m']$ we have that

$$\begin{aligned}
&\mathrm{H}(\widetilde{Y}_i \mid \widetilde{X}_{1,\ldots,i-1}) \qquad (4)\\
&\qquad \geq \mathrm{H}(Y_{J+i-1 \bmod m} | X_1,\ldots,X_{(J+i-1 \bmod m)-1})\\
&\qquad = \mathop{\mathrm{E}}_{i' \xleftarrow{\mathrm{R}} [m]}[\mathrm{H}(Y_{i'} \mid X_1,\ldots,X_{i'-1})].
\end{aligned}$$

Let $\widetilde{D}$ be an adversary the violates the next-block pseudoentropy of $\widetilde{X}$. We define D for breaking the next-block pseudoentropy of X as follows: on input $(x_1,\dots,x_{i-1},z)$, D generates a random sample $x' = (x'_1,\dots,x'_{m'})$ from $\widetilde{X}$ (using $O_{X,Y}$). It then selects $i' \in [m']$ uniformly at random such that $i' = j + i - 1 \bmod m$, where j is the value of J in the generation of x', and returns $\widetilde{D}^{O_{\widetilde{X},\widetilde{Y}}}(x'_1,\dots,x'_{i'-i},x_1,\dots,x_{i-1},z)$, while answering $\widetilde{D}$'s queries to $O_{\widetilde{X},\widetilde{Y}}$ using $O_{X,Y}$.

We note that D makes at most ℓ times more oracle queries than $\widetilde{D}$, and that D can be implemented in the running time of $\widetilde{D}$ plus $O(\ell\cdot m\cdot\log|\mathcal{U}|)$.

For every Y as above with $\sum_{i\in[m]} \mathrm{H}(Y_i \mid X_{1,\dots,i-1}) \geq k$, Equation (4) yields that $\widetilde{Y}$ is a random variable that $\widetilde{D}$ should be able to next-block distinguish from $\widetilde{X}$. Since the query to $\widetilde{D}$ done by D is distributed identically to a random challenge to $\widetilde{D}$ with respect to the joint distribution $(\widetilde{X},\widetilde{Y})$, it follows that

$$\delta^{\widetilde{D}}_{\widetilde{X},\widetilde{Y}} = \delta^{D}_{X,Y} \leq L\cdot\varepsilon = \widetilde{L}\cdot(\ell\varepsilon),$$

where L and $\widetilde{L}$ are the number of oracle calls made by D and $\widetilde{D}$, respectively.This is contradiction to the assumption about the next-block pseudoentropy of X. □

5.2 Next-block Pseudoentropy Converts to Pseudo-Min-Entropy

In this section we show how to transform next-block (Shannon) pseudoentropy to next-block pseudo-min-entropy, while increasing the overall entropy gap. The transformation of X is simply a t-fold parallel repetition of X (i.e., every block of the new random variable X^t is composed of t corresponding blocks of t independent copies of X). This generalizes an analogous transformation that was used by Håstad et al. [14] in the context of standard (i.e. single-block) pseudoentropy.

Given an m-block random variable V taking values in $\mathcal{U}^m$ and an integer $t > 0$, we let $V^t = ((V_1^{(1)},\dots,V_1^{(t)}),\dots,(V_m^{(1)},\dots,V_m^{(t)})) \in (\mathcal{U}^t)^m$, where the $V^{(i)}$'s are iid copies of V.

Lemma 5.3. *Let n be a security parameter, and $m = m(n) = \mathrm{poly}(n)$, $t = t(n) = \mathrm{poly}(n)$, be $\mathrm{poly}(n)$-time computable functions, and let X be a random variable over $\mathcal{U}^m$ where every block of X has (T,ε) next-block pseudoentropy at least α, for $T = T(n)$, $\varepsilon = \varepsilon(n)$, $\alpha = \alpha(n)$. Then for every $\kappa = \kappa(n) > 0$ it holds that every block of X^t has (T',ε') next-block pseudo-*min-*entropy α', where*

- $T' = T'(n) = T - O(m\cdot t\cdot\log|\mathcal{U}|)$.
- $\varepsilon' = \varepsilon'(n) = t^2\cdot(\varepsilon + 2^{-\kappa} + 2^{-c\cdot t})$ *for a universal constant $c > 0$, and*
- $\alpha' = \alpha'(n) = t\cdot\alpha - \Gamma(t,\kappa,|\mathcal{U}|)$, *for $\Gamma(t,\kappa,|\mathcal{U}|) \in O(\sqrt{t\cdot\kappa}\cdot\log(|\mathcal{U}|\cdot t))$.*

Furthermore, the reduction between the security of X^t and X is fully black-box.

Notice that the $t\cdot\alpha$ term is the largest we could hope for the pseudoentropy — getting α bits of pseudoentropy per copy. However, since we wish to move from a pseudo-form of Shannon entropy (measuring randomness on average) to a pseudo-form of min-entropy (measuring randomness with high probability), we may have a deviation that grows like $\sqrt{t}$. By taking t large enough, this deviation becomes insignificant.

In more detail, consider the case that X has next-block pseudoentropy at least α with polynomial security, i.e. T and $1/\varepsilon$ can be taken to be arbitrarily large polynomials in n, and we would like to deduce that X^t has next-block pseudo-min-entropy α' with polynomial security. Moreover, assume

$\mathcal{U} = \{0,1\}$. Then we should take κ and t to be an arbitrarily large multiples of $\log n$, and we have $\alpha' = t \cdot (\alpha - O(\sqrt{(\log n)/t}) \cdot \log t)$. So if we would like to have pseudo-min-entropy at least $t \cdot (\alpha - \delta)$, we should take t to be $\mathrm{polylog}(n)/\delta^2$. In our application, we have $\delta = \Theta(\Delta/n) = \Theta(\log n/n)$, so we take $t = \tilde{O}(n^2)$ copies.

Proof. For a random variables Y_i over $\mathcal{U}$ jointly distributed with X, we defined the, jointly distributed with X^t, variable $(Y_i)^t = ((Y_i)^{(1)}, \dots, (Y_i)^{(t)})$, where for each $j \in [t]$ the entry $(Y_i)^{(j)}$ is jointly distributed with the entry $X^{(j)}$ in X^t according to the joint distribution (X, Y_i). Given an adversary D_t that violates the next-block pseudo-min-entropy of X^t, we define D for breaking the next-block pseudoentropy of X as follows: given an input $(x_1, \dots, x_{i-1}, z)$ and oracle access to $O_{X,Y}$, D first samples $j \xleftarrow{\mathrm{R}} [t]$ and (x^t, y^t) from $(X^t, (Y_i)^t)$ (using $O_{X,Y}$). Then it replaces $(x^t_{j,1} \dots, x^t_{j,i-1})$ (the $(i-1)$ prefix of j'th column of x^t) with $(x_1, \dots, x_{i-1})$, sets $z^{[j]} = (x^t_{i,1}, \dots, x^t_{i,j-1}, z, y^t_{j+1}, \dots, y^t_t)$, and returns $D_t^{O_{X^t,(Y_i)^t}}(x^t_{1,\dots,i-1}, z^{[j]})$, while answering D_t's queries to $O_{X^t,(Y_i)^t}$ using $O_{X,Y}$. That is, D queries D_t on the j'th hybrid between $(X^t_{1,\dots,i-1}, (Y_i)^t)$ and $X^t_{1,\dots,i}$.

We note that D makes at most t times more oracle calls than D_t, and that it can be implemented in the same time as D plus $O(t \cdot m \cdot \log |\mathcal{U}|)$.

Assuming that $\mathrm{H}(Y_i \mid X_{1,\dots,i-1}) \geq \alpha$ for $i \in [m]$, Lemma 2.1 yields that there exists a random variable W over $\mathcal{U}^t$ jointly distributed with X^t, such that the following hold:

1. $\Delta((X^t_{1,\dots,i-1}, (Y_i)^t), (X^t_{1,\dots,i-1}, W)) \leq 2^{-\kappa} + 2^{-c \cdot t}$ for a universal constant $c > 0$, and
2. $\mathrm{H}_\infty(W \mid x_{1,\dots,i-1}) \geq \alpha - \Gamma(t, \kappa, |\mathcal{U}|)$ for every $x \in \mathrm{Supp}(X^t)$.

For $j \in [t]$, let $Z^{[j]} = ((X^t_i)_{1,\dots,j}, (Y_i)^t_{j+1,\dots,t})$ (i.e., the j'th hybrid between X^t_i and $(Y_i)^t$). It follows that for each $i \in [m]$, we have

$$
\begin{aligned}
&\delta^D_{X,Y,i} \\
&:= \frac{1}{t} \cdot \sum_{j \in [t]} \big(\Pr[D_t^{O_{X^t,(Y_i)^t}}(X^t_{1,\dots,i-1}, Z^{[j]}) = 1] - \Pr[D_t^{O_{X^t,(Y_i)^t}}(X^t_{1,\dots,i-1}, Z^{[j-1]}) = 1]\big) \\
&= \frac{1}{t} \cdot \big(\Pr[D_t^{O_{X^t,(Y_i)^t}}(X^t_{1,\dots,i-1}, Z^{[t]}) = 1] - \Pr[D_t^{O_{X^t,(Y_i)^t}}(X^t_{1,\dots,i-1}, Z^{[0]}) = 1]\big) \\
&= \frac{1}{t} \cdot \big(\Pr[D_t^{O_{X^t,(Y_i)^t}}(X^t_{1,\dots,i-1}, X^t_i) = 1] - \Pr[D_t^{O_{X^t,(Y_i)^t}}(X^t_{1,\dots,i-1}, (Y_i)^t) = 1]\big) \\
&\geq \frac{1}{t} \cdot \big(\Pr[D_t^{O_{X^t,W}}(X^t_{1,\dots,i-1}, X^t_i) = 1] - \Pr[D_t^{O_{X^t,W}}(X^t_{1,\dots,i-1}, W) = 1] - L' \cdot (2^{-\kappa} - 2^{-c \cdot t})\big) \\
&=: \frac{1}{t} \cdot \big(\delta^{D_t}_{X^t,W,i} - L' \cdot (2^{-\kappa} - 2^{-c \cdot t})\big),
\end{aligned}
$$

where L' is a bound on the number of oracles calls made by D_t. Taking expectations over $i \xleftarrow{\mathrm{R}} [m]$, we have

$$
\begin{aligned}
\delta^D_{X,Y} &\geq \frac{1}{t} \cdot \Big(\delta^{D_t}_{X^t,W} - L' \cdot (2^{-\kappa} + 2^{-c \cdot t})\Big) \\
&\geq \frac{1}{t} \cdot \big(t^2 \cdot L' \cdot (\varepsilon + 2^{-\kappa} + 2^{-c \cdot t}) - L' \cdot (2^{-\kappa} + 2^{-c \cdot t})\big) \\
&\geq L' \cdot t \cdot \varepsilon = L\varepsilon,
\end{aligned}
$$

where $L = t \cdot L'$ is a bound on the number of oracle calls made by D. Since this holds for all $\{Y_i\}$ such that $\mathrm{H}(Y_i \mid X_{1,\dots,i-1}) \geq \alpha$, this contradicts the assumption about the next-block pseudoentropy of X. □

5.3 Next-block Pseudo-Min-Entropy to Pseudorandomness

For our final step, we assume that X is such that each of the m blocks of X has large next-block pseudo-min-entropy α. Using a two-universal independent hash function S, we extract almost all its pseudo-min-entropy of each block individually. The result is a sufficiently long pseudorandom bit sequence. This is a computational analogue of block-source extraction in the literature on randomness extractors [2, 29].

Lemma 5.4. *Let n be a security parameter, $m = m(n) = \mathrm{poly}(n)$, $t = t(n) = \mathrm{poly}(n)$, $\alpha = \alpha(n) \in [t(n)]$ and $\kappa = \kappa(n) \in [\alpha(n)]$ be $\mathrm{poly}(n)$-time computable integer functions. There exists an efficient procedure $\mathrm{Ext} \in \mathrm{NC}^1$ that on input $x \in (\{0,1\}^t)^m$ and $s \in \{0,1\}^t$, outputs a string $y \in \{0,1\}^{t+m\cdot(\alpha-\kappa)}$ such that the following holds.*

Let X be a random variable over $(\{0,1\}^t)^m$ such that every block of X has (T,ε) next-block pseudo-min-entropy α, for $T = T(n)$ and $\varepsilon = \varepsilon(n)$, then $\mathrm{Ext}(X, U_t)$ is $(T - m \cdot t^{O(1)}, m \cdot (\varepsilon + 2^{-\kappa/2}))$ pseudorandom. Moreover, the reduction between the pseudorandomness of $\mathrm{Ext}(X, U_t)$ and the security X is fully black-box.

Proof. Let $\mathrm{Ext}(x, s) := (s, s(x_1), \dots, s(x_m))$, where s is interpreted as a member of a family of two-universal hash functions from t bits to $\alpha - \kappa$ bits in NC^1 (such as $s(x) := s \cdot x$ over $GF(2^t)$ truncated to $\alpha - \kappa$ bits). Let D_{PRG} be an adversary for the pseudorandomness of $\mathrm{Ext}(X, U_t)$, and let δ_{PRG} be its distinguishing advantage. We define D for breaking the next-block pseudoentropy of X as follows: on input $(x_1, \dots, x_{i-1}, z)$, D returns $D_{\mathrm{PRG}}(s, s(x_1), \dots, s(x_{i-1}), s(z), U_{(\alpha-\kappa)\cdot(m-i)})$, where s is uniformly chosen from $\{0,1\}^t$.

We note that D makes no oracle calls (and thus we count its query complexity as one), and that its running-time is at most that of D_{PRG} plus $m \cdot \mathrm{poly}(t)$.

Let $Z^{[i]}(W) := (S, S(X_1), \dots, S(X_{i-1}), S(W), U_{(\alpha-\kappa)\cdot(m-i)})$, for a uniformly distributed hash function S. Let $Y = \{Y_1, \dots, Y_m\}$ be a set of random variable over $\mathcal{U}$ jointly distributed with X, with $\mathrm{H}_\infty(Y_i \mid X_{1,\dots,i-1} = x_{1,\dots,i-1}) \geq \alpha$ for every $x \in \mathrm{Supp}(X)$ and $i \in [m]$. The Leftover Hash Lemma [20, 19] yields that $Z^{[i]}(Y_i)$ has statistical difference at most $2^{-\kappa/2}$ from $Z^{[i-1]}(X_{i-1})$. Thus

$$\begin{aligned}
\delta_{\mathrm{PRG}} &= \Pr[D_{\mathrm{PRG}}(Z^{[m]}(X_m)) = 1] - \Pr[D_{\mathrm{PRG}}(Z^{[0]}(X_0)) = 1] \\
&= \sum_{i=1}^{m} \Big(\Pr[D_{\mathrm{PRG}}(Z^{[i]}(X_i)) = 1] - \Pr[D_{\mathrm{PRG}}(Z^{[i-1]}(X_{i-1})) = 1] \Big) \\
&\leq \sum_{i=1}^{m} \Big(\Pr[D_{\mathrm{PRG}}(Z^{[i]}(X_i)) = 1] - \Pr[D_{\mathrm{PRG}}(Z^{[i]}(Y_i)) = 1] + 2^{-\kappa/2} \Big) \\
&= m \cdot (\delta^D_{X,Y} + 2^{-\kappa/2}) \\
&\leq m \cdot (\varepsilon + 2^{-\kappa/2}).
\end{aligned}$$

□

5.4 Putting It Together

We are now ready to prove Theorem 5.1.

Theorem 5.5 (Theorem 5.1, restated)**.** *Let n be a security parameter, and let $m = m(n)$, $\Delta = \Delta(n) \in [1/\operatorname{poly}(n), n]$, and $\kappa = \kappa(n) \in \{1, \ldots, n\}$ be $\operatorname{poly}(n)$-time computable. For every polynomial-time computable, m-block generator $G_{nb}\colon \{0,1\}^n \mapsto \{0,1\}^m$, there exists a polynomial-time computable generator $G : \{0,1\}^d \to \{0,1\}^{d\cdot(1+\Omega(\Delta/n))}$ with seed length*

$$d = d(n) = O\left(\frac{n^2 \cdot m^2 \cdot \kappa \cdot \log^2 n}{\Delta^3}\right)$$

such that the following holds:

Security: *If G_{nb} has (T, ε)-next-block pseudoentropy at least $n + \Delta$, for $T = T(n), \varepsilon = \varepsilon(n)$, then G is a $(T - n^{O(1)}, n^{O(1)} \cdot (\varepsilon + 2^{-\kappa}))$-pseudorandom generator. Moreover, the reduction is fully black-box.*

Complexity: *G is computable in NC^1 with $O(d/n)$ (uniformly random) oracle calls to G_{nb}.*

Proof. Let $X = G_{nb}(U_n)$. Without loss of generality, we may assume that the number m of output blocks (=bits) of G_{nb} is a power of 2 (by padding with zeroes if necessary). X can be generated using n random bits, and has next-block pseudoentropy $k = n + \Delta$.

We now set $\ell = \lceil 2(n + \Delta + \log m)/\Delta \rceil = O(n/\Delta)$ and apply Entropy Equalization (Lemma 5.2) to obtain $\widetilde{X} := \operatorname{Eq}(J, X^{(1)}, \ldots, X^{(\ell)})$, where J is uniformly distributed over $[m]$ and the $X^{(i)}$'s are iid copies of X. $\widetilde{X}$ can be generated using $d_\ell = \log m + \ell \cdot n$ random bits, and has $m' = (\ell - 1) \cdot m$ blocks. Every block of $\widetilde{X}$ has $(T_\ell = T - O(\ell \cdot m), \varepsilon_\ell = \ell \cdot \varepsilon)$-next-block pseudoentropy at least $\alpha_\ell = k/m$. Thus the *total* next-block pseudoentropy in $\widetilde{X}$ is at least

$$\begin{aligned}
m' \cdot \alpha_\ell &= m \cdot (\ell - 1) \cdot k/m \\
&= (n + \Delta) \cdot (\ell - 1) \\
&\geq n \cdot \ell + \log m + \Delta\ell/2 \\
&= d_\ell + \Delta\ell/2,
\end{aligned}$$

where the inequality follows from the setting of ℓ.

Next we apply t-fold parallel repetition (Lemma 5.3) to obtain $(\widetilde{X})^t$, for a parameter $t = \operatorname{poly}(n)$ to be set below. $(\widetilde{X})^t$ can be generated using $d_t = t \cdot d_\ell$ random bits and has m' blocks of t bits each. Every block of $(\widetilde{X})^t$ has $(T_t = T_\ell - O(m' \cdot t), \varepsilon_t = t^2 \cdot (\varepsilon_\ell + 2^{-\kappa} + 2^{-\Omega(t)}))$-next-block pseudo-*min*-entropy $\alpha_t = t \cdot \alpha_\ell - \Gamma(t, \kappa)$.

Finally, we apply Lemma 5.4, extracting $\alpha_t - 2\kappa$ bits from each block of $(\widetilde{X})^t$ using a seed of length t. This yields our final output $\operatorname{Ext}((\widetilde{X})^t, U_t)$, which is (T', ε') pseudorandom, for

$$\begin{aligned}
T' &= T_t - m' \cdot t^{O(1)} = T - \operatorname{poly}(n) \\
\varepsilon' &= m' \cdot (\varepsilon_t + 2^{-\kappa}) = \operatorname{poly}(n) \cdot (\varepsilon + 2^{-\kappa} + 2^{-\Omega(t)}).
\end{aligned}$$

$\operatorname{Ext}((\widetilde{X})^t, U_t)$ can be generated using a seed of length

$$d = d_t + t = t \cdot d_\ell + t = O(t \cdot \ell \cdot n),$$

and has an output length of

$$\begin{aligned} d' &= m' \cdot (\alpha_t - \kappa) + t \\ &= m' \cdot (t \cdot \alpha_\ell - \Gamma(t,\kappa) - \kappa) + t \\ &\geq t \cdot d_\ell + t\Delta\ell/2 - m' \cdot (\Gamma(t,\kappa) + \kappa) + t \\ &= d + t\Delta\ell/2 - m' \cdot O(\sqrt{t\kappa} \cdot \log t + \kappa) \\ &\geq d + t\Delta\ell/4 = (1 + \Omega(\Delta/n)) \cdot d, \end{aligned}$$

where the last inequality follows from an appropriate setting of

$$t = O\left(\left(\frac{m'}{\Delta\ell}\right)^2 \cdot \kappa \cdot \log^2\left(\frac{m'\kappa}{\Delta\ell}\right)\right) = O\left(\frac{m^2 \cdot \kappa \cdot \log^2 n}{\Delta^2}\right).$$

Finally, we can bound the seed length d by

$$d = O(t \cdot n \cdot \ell) = O\left(\frac{n^2 \cdot m^2 \cdot \kappa \cdot \log^2 n}{\Delta^3}\right).$$

□

6 Deducing the Main Results

We are now ready to prove the main result of the paper.

Theorem 6.1 (Pseudorandom generators from one-way functions)**.** *Let n be a security parameter and $f : \{0,1\}^n \mapsto \{0,1\}^n$ a polynomial-time computable function. For all* poly(n)*-time computable functions $\varepsilon = \varepsilon(n) \leq 1/n^c$ (where c is a universal constant) and $\kappa = \kappa(n) \in [n/4]$, there exists an efficient generator G from strings of length $d = d(n) = O(n^4 \cdot \kappa \cdot \log^2 n / \log^3(1/\varepsilon))$ to strings of length $d \cdot (1 + \Omega(\log(1/\varepsilon))/n)$, such that the following holds:*

Security: *Assume that f is a (T,ε) one-way function for $T = T(n)$. Then for every* poly(n)*-time computable function $\varepsilon' = \varepsilon'(n) \geq 2^{-\kappa}$, G is a $(T \cdot (\varepsilon'/n)^{O(1)}, \varepsilon' \cdot n^{O(1)})$-pseudorandom generator. Moreover, the reduction is fully black-box,*

Complexity: *G is computable in* NC^1 *with $O(d/n)$ (uniformly random) oracle calls to f.*

Proof. We let c be the constant of Theorem 4.9. We start by applying Theorem 4.9 on f, to get a generator $G_{nb} \colon \{0,1\}^{d_1} \mapsto \{0,1\}^{m_1}$ that has $(T \cdot (\varepsilon'/n)^{O(1)}, \varepsilon')$-next-block-pseudoentropy $k = d_1 + \log(1/\varepsilon)/2$, where $d_1, m_1 \in O(n)$.

In the next step we apply Theorem 5.1 with respect to the above G_{nb}, κ and $\Delta := \log(1/\varepsilon)/2$, to get an efficient generator from strings of length

$$d = O\left(\frac{d_1^2 \cdot m_1^2 \cdot \kappa \cdot \log^2 n}{\Delta^3}\right) = O\left(\frac{n^4 \cdot \kappa \cdot \log^2 n}{\log^3(1/\varepsilon)}\right)$$

to strings of length $d \cdot (1 + \Omega(\log(1/\varepsilon))/n)$, that is $(T \cdot (\varepsilon'/n)^{O(1)} - n^{O(1)}, (\varepsilon_1 + 2^{-\kappa}) \cdot n^{O(1)}) = (T \cdot (\varepsilon'/n)^{O(1)}, \varepsilon' \cdot n^{O(1)})$-pseudorandom. □

The above theorem yields the following important corollaries.

Corollary 6.2 (Pseudorandom generator from one-way functions — polynomial security case)**.** *Let n be a security parameter and $f\colon \{0,1\}^n \mapsto \{0,1\}^n$ a one-way function.Then there exists a pseudorandom generator G from strings of length $d = d(n) \in \tilde{O}(n^4)$ to strings of length $d \cdot (1 + \Omega((\log n)/n))$.*

Furthermore, the reduction is fully black-box. G is computable in NC^1 with $O(d/n)$ (uniformly random) oracle calls to f.

Proof. Applying Theorem 6.1 on f, $\varepsilon = 1/n^c$ and $\kappa = \log^2 n$, we get an efficient generator G of the stated input and output lengths. Assume now that G is not a pseudorandom generater. Namely, there exists $p \in \mathrm{poly}$ such that G is *not* $(p(n), 1/p(n))$ pseudorandom. Therefore, G is not $(T \cdot (\varepsilon'/n)^{O(1)}, \varepsilon' \cdot \mathrm{poly}(n))$-pseudorandom, for $\varepsilon' := 1/p(n) \cdot \mathrm{poly}(n) > 2^{-\kappa}$ and $T = \mathrm{poly}(n) \cdot p(n)$. It follows that f is not (T, ε)-one-way, in contradiction. □

Corollary 6.3 (Pseudorandom generator from one-way functions —exponential hardness case)**.** *Let $f\colon \{0,1\}^n \mapsto \{0,1\}^n$ be a $(2^{\Omega(n)}, 2^{-\Omega(n)})$-one-way function. Then*

1. *There exists a $(2^{\Omega(n)}, 2^{-\Omega(\log^2 n)})$-pseudorandom generator G from strings of length $d = d(n) \in \tilde{O}(n)$ to strings of length $d \cdot (1 + \Omega(1))$, and*

2. *There exists a $(2^{\Omega(n)}, 2^{-\Omega(n)})$-pseudorandom generator G from strings of length $d = d(n) \in \tilde{O}(n^2)$ to strings of length $d \cdot (1 + \Omega(1))$,*

Furthermore, in both cases it holds that the reduction is fully black-box, G is computable in NC^1 with $O(d/n)$ (uniformly random) oracle calls to f.

Proof. Immediate from Theorem 6.1, taking $\kappa = \log^2(n)$ and $\kappa \in \Omega(n)$ in the first and second cases respectively, and $\varepsilon' = 2^{-\kappa}$. □

Acknowledgments

We thank Benny Applebaum, Oded Goldreich, Thomas Holenstein, and Emanuele Viola for very helpful conversations.

References

[1] B. Applebaum, Y. Ishai, and E. Kushilevitz. Cryptography in NC^0. *SIAM Journal on Computing*, 36, 2006.

[2] B. Chor and O. Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM J. Comput.*, 17(2):230–261, Apr. 1988.

[3] U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1 (2):77–94, 1988.

[4] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 25–32, 1989.

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

[6] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(1):691–729, 1991.

[7] O. Goldreich, H. Krawczyk, and M. Luby. On the existence of pseudorandom generators. *SIAM Journal on Computing*, 22(6):1163–1175, 1993.

[8] V. Guruswami and M. Sudan. List decoding algorithms for certain concatenated codes. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, pages 181–190 (electronic), New York, 2000. ACM. doi: 10.1145/335305.335327. URL http://dx.doi.org/10.1145/335305.335327.

[9] I. Haitner, O. Reingold, S. Vadhan, and H. Wee. Inaccessible entropy. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC).*

[10] I. Haitner, D. Harnik, and O. Reingold. On the power of the randomized iterate. In *Advances in Cryptology – CRYPTO 2006*, 2006.

[11] I. Haitner, D. Harnik, and O. Reingold. Efficient pseudorandom generators from exponentially hard one-way functions. In *Automata, Languages and Programming, 24th International Colloquium, ICALP*, 2006.

[12] I. Haitner, M. Nguyen, S. J. Ong, O. Reingold, and S. Vadhan. Statistically hiding commitments and statistical zero-knowledge arguments from any one-way function. *SIAM Journal on Computing*, 39(3):1153–1218, 2009.

[13] I. Haitner, O. Reingold, S. Vadhan, and H. Wee. Inaccessible entropy. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC '09)*, pages 611–620, 31 May–2 June 2009.

[14] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[15] T. Holenstein. Key agreement from weak bit agreement. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 664–673, 2005.

[16] T. Holenstein. Pseudorandom generators from one-way functions: A simple construction for any hardness. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006*, 2006.

[17] T. Holenstein. Strengthening key agreement using hard-core sets - PhD thesis, 2006.

[18] R. Impagliazzo. A personal view of average-case complexity. In *Proceedings of the Tenth Annual Structure in Complexity Theory Conference*, pages 134–147. IEEE Computer Society, 1995.

[19] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 248–253, 1989.

[20] R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random generation from one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 12–24. ACM Press, 1989.

[21] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, 1988.

[22] M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

[23] A. A. Razborov and S. Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55 (1):24–35, Aug. 1997.

[24] O. Reingold, L. Trevisan, and S. P. Vadhan. Notions of reducibility between cryptographic primitives. In *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004*, pages 1–20.

[25] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62:236–266, 2001.

[26] S. P. Vadhan. Constructing locally computable extractors and cryptosystems in the bounded-storage model. *Journal of Cryptology*, 17(1):43–77, January 2004.

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

[28] A. C. Yao. Protocols for secure computations. pages 160–164, 1982.

[29] D. Zuckerman. Simulating BPP using a general weak random source. *Algorithmica*, 16(4/5): 367–391, Oct./Nov. 1996.

A The Uniform Hardcore Lemma

In this section, we show how to deduce our version of the Uniform Hardcore Lemma (Proposition 4.7) from what Holenstein proves. Holenstein's statement of the Hardcore Lemma requires the functions $\gamma(n)$ and $\delta(n)$ to be computable in time $\mathrm{poly}(n)$ and to be at least $1/\mathrm{poly}(n)$. However, the same proof yields the following, where we give δ and γ as input to the algorithm M, and we allow a loss of $(\delta\gamma)^{O(1)}$ in the running time:

Proposition A.1 ([17], Thm 6.8)**.** *Let n be a security parameter, and let $h\colon \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}$ and $V\colon \{0,1\}^n \mapsto \{0,1\}$ be polynomial-time computable functions. Let $\delta = \delta(n) \in [\delta_0, 1]$, and $\gamma = \gamma(n) \in [0,1] > 2^{-n/3}$. Assume that*

$$\Pr[M(\delta, \gamma, h(U_n)) = V(U_n)] \leq 1 - \delta/2 + \gamma^2\delta^5/8192$$

for every probabilistic algorithm M running in time $T = T(n)$ and large enough n. Then for every oracle-aided predictor P running in time $T \cdot (\gamma\delta/n)^{O(1)}$ and all sufficiently large n, there exists a set $L \subseteq \{0,1\}^n$ of density at least δ such that

$$\Pr_{W \overset{R}{\leftarrow} L}[P^{\chi_L}(h(W)) = V(W)] < (1+\gamma)/2,$$

where χ_L is the characteristic function of L, provided that all the queries of P to χ_L are computed independently of the input $h(W)$. Furthermore, the reduction is fully black-box.

Note the hypothesis of Holenstein's lemma is slightly stronger in that it works even if the success probability of M is slightly higher than $1 - \delta/2$; this slackness will be convenient in deducing Proposition 4.7, which we now do.

Proof of Proposition 4.7. Assume that

$$\Pr[M(\delta_0, \gamma, h(U_n)) = V(U_n)] \leq 1 - \delta/2 \tag{5}$$

for every probabilistic algorithm M running in time $T = T(n)$ and large enough n. Then we will argue that

$$\Pr[M'(\delta', \gamma, h(U_n)) = V(U_n)) \leq 1 - \delta'/2 + \alpha, \tag{6}$$

for every probabilistic algorithm M' running in time $T' = T(n) \cdot (\delta_0\gamma)^{O(1)}$ and large enough n, where δ' is the smallest multiple of $\alpha = \delta_0^5\gamma^2/8192$ that is at least as large as δ (so $\delta' \leq \delta + \alpha$). Suppose for contradiction that there is an M' violating Inequality (6). Then we can obtain an M violating Inequality (5) by trying $\delta' = i \cdot \alpha$ for each $i = 0, \ldots, \lfloor 1/\alpha \rfloor$, estimating the success probability to within accuracy $\alpha/3$ (by random sampling), and finally running M' with the value of δ' that achieves maximum estimated success probability.

Since $\alpha \leq (\delta')^5\gamma^2/8192$, we can apply Proposition A.1 to obtain, for every predictor P running in time $T' \cdot (\gamma\delta'/n)^{O(1)} = T \cdot (\gamma\delta_0/n)^{O(1)}$, a hardcore set L of density $\delta' \geq \delta$. □