Electronic Colloquium on Computational Complexity, Report No. 84 (2017)

The Many Entropies in One-Way Functions

Iftach Haitner and Salil Vadhan

Abstract Computational analogues of information-theoretic notions have given rise to some of the most interesting phenomena in the theory of computation. For example, *computational indistinguishability*, Goldwasser and Micali [9], which is the computational analogue of statistical distance, enabled the bypassing of Shanon's impossibility results on perfectly secure encryption, and provided the basis for the computational theory of pseudorandomness. *Pseudoentropy*, Håstad, Impagliazzo, Levin, and Luby [17], a computational analogue of entropy, was the key to the fundamental result establishing the equivalence of pseudorandom generators and one-way functions, and has become a basic concept in complexity theory and cryptography. This tutorial discusses two rather recent computational notions of entropy, both of which can be easily found in any one-way function, the most basic cryptographic primitive. The first notion is *next-block pseudoentropy*, Haitner, Reingold, and Vadhan [14], a refinement of of pseudoentropy that enables simpler and more efficient construction of pseudorandom generators. The second is *inaccessible entropy*, Haitner, Reingold, Vadhan, and Wee [11], which relates to *unforgeability* and is used to construct simpler and more efficient universal one-way hash functions and statistically hiding commitments.

Iftach Haitner
School of Computer Science, Tel Aviv University. E-mail: `iftachh@cs.tau.ac.il`, member of the Israeli Center of Research Excellence in Algorithms (ICORE) and the Check Point Institute for Information Security. Research supported by ERC starting grant 638121.

Salil Vadhan
John A. Paulson School of Engineering & Applied Sciences, Harvard University. E-mail: `salil@seas.harvard.edu`. Written while visiting the Shing-Tung Yau Center and the Department of Applied Mathematics at National Chiao-Tung University in Hsinchu, Taiwan. Supported by NSF grant CCF-1420938 and a Simons Investigator Award.

ISSN 1433-8092

Contents

1 Introduction

One-way functions (OWFs), functions that are easy to compute and hard to invert, are the most basic, unstructured form of cryptographic hardness [22]. Yet, in a sequence of celebrated results, mostly in the 1980s and early 1990s, one-way functions were shown to imply a rich collection of cryptographic schemes and protocols, such as digital signatures and secret-key encryption schemes. At the basis of this beautiful mathematical structure are a few constructions of basic primitives: pseudorandom generators (Håstad et al. [17]), universal one-way hash functions (Naor and Yung [26], Rompel [27]), and more recently, statistically hiding commitment schemes (Haitner, Nguyen, Ong, Reingold, and Vadhan [10]). These powerful plausibility results shape our understanding of hardness, secrecy, and unforgeability in cryptography. For instance, the construction of pseudorandom generators provides strong evidence that computationally secure encryption is much richer than information-theoretically secure encryption, as it allows encrypting many more bits than the key length, in contrast to Shannon's impossibility result for information-theoretic security [28]. The construction of universal one-way hash functions yields that some "public-key" objects, such as signature schemes, can be built from "private-key" primitives, like one-way functions. A recent line of results [11, 12, 14, 29] simplified and improved all of these constructions. The crux of each new construction is defining the "right" notion of *computational entropy* and recovering this form of entropy from one-way functions.

Computational entropy. Computational analogues of information-theoretic notions have given rise to some of the most interesting phenomena in the theory of computation. For example, *computational indistinguishability*, a computational analogue of statistical indistinguishability introduced by Goldwasser and Micali [9], enabled the bypassing of Shannon's impossibility results on perfectly secure encryption [28], and provided the basis for the computational theory of pseudorandomness [2, 32]. *Pseudoentropy*, a computational analogue of entropy introduced by Håstad et al. [17], was the key to their fundamental result establishing the equivalence of pseudorandom generators and one-way functions, and has become a basic concept in complexity theory and cryptography. The above notions were further refined in [14, 29], and new computational analogues of entropy to quantify unforgeability were introduced in [11, 12]. These new abstractions have led to much simpler and more efficient constructions based on one-way functions, and to a novel equivalence between (parallelizable) constant-round statistical zero-knowledge arguments and constant-round statistically hiding commitments.

The purpose of this tutorial is to explain these computational notions of entropy and their application in constructing cryptographic primitives. The utility of the computational notions of entropy is to bridge between the very unstructured form of hardness of the primitive we start with (e.g., one-wayness) and the typically much more structured form of hardness that appears in the primitive we are trying to construct. The benefit of using such computational notions of entropy is that there exists well-developed machinery for manipulating information-theoretic entropy and making it more structured (e.g., through taking many independent copies and applying hash functions and randomness extractors); with care, analogous tools can be applied to the computational notions. For example, in each of the two constructions presented in this tutorial, the first step is to construct a "generator" with a noticeable gap between its real output entropy and its computational entropy—entropy from the point of view of a computationally bounded adversary. (For each construction, we use a different notion computational entropy.) The next step is to increase the gap between real and computational entropy and to convert them into worst-case analogues (e.g., min-entropy and max-entropy) using the standard information-theoretic tools of taking many independent samples. Finally, hashing and randomness extractors are used to obtain more structured randomness generators.

In the following, we discuss the two major types of computational entropy notions that can be found in any one-way function. *pseudoentropy*, which comes to quantify pseudorandomness and *secrecy*, and *inaccessible entropy*, which comes to quantify *unforgeability*. We do that while focusing on *next-block pseudoentropy*, a refinement of the traditional notion of pseudoentropy, and on the type of inaccessible entropy that is related to, and used as in intermediate step to construct, statistically hiding commitment schemes. In the main body of this tutorial, we discuss these two notions further, and exemplify their usability with applications to one-way function based primitives.

1.1 Pseudoentropy

A random variable X over $\{0,1\}^n$ is *pseudorandom* if it is computationally indistinguishable from U_n.[1] The most natural quantitative variant of pseudorandomness is the so-called *HILL pseudoentropy* (stands for Håstad, Impagliazzo, Levin, and Luby), or just pseudoentropy.

Definition 1 ((HILL) pseudoentropy, [17], informal). *A random variable X is said to have* 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.*[2]

A function (i.e., a generator) $G : \{0,1\}^n \mapsto \{0,1\}^{m(n)}$ has pseudoentropy k if $G(U_n)$ has pseudoentropy k. An efficiently computable $G : \{0,1\}^n \mapsto \{0,1\}^{m(n)}$ is a pseudoentropy generator *if it has pseudoentropy (at least) $\mathrm{H}(G(U_n))) + \Delta(n)$ for some $\Delta(n) \geq 1/\operatorname{poly}(n)$. We refer to Δ as the* entropy gap *of G.*[3]

Pseudoentropy plays a key role in the Håstad et al. [17] construction of pseudorandom generators from one-way functions. A pseudorandom generator (PRG) is an efficient length-extending function whose output distribution, over uniformly chosen input, is pseudorandom. Note that every pseudorandom generator $G\colon \{0,1\}^n \mapsto \{0,1\}^{m(n)}$ is a pseudoentropy generator with entropy gap at least $m(n) - n$; take $Y = U_{m(n)}$ and note 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 even with $\mathrm{H}(Y) \ll n$ (as long as $\mathrm{H}(G(U_n))$ is even smaller). Yet, Håstad et al. [17] showed that also the converse is true, using pseudoentropy generators to construct pseudorandom generators. The first and key step of their main result (that one-way functions imply pseudorandom generators) was to show that a simple modification of any one-way function is a pseudoentropy generator with small but noticeable entropy gap, where the rest of their construction is "purifying" this generator's pseudoentropy into pseudorandomness, and thus turning it into a PRG. This shows in a sense that (a simple modification of) one-way functions have the computational notion of entropy that pseudorandom generators take to the extreme.

Constructing pseudoentropy generator from an *injective* one-way function is easy. Given such an injective function $f\colon \{0,1\}^n \mapsto \{0,1\}^*$, let $G(x) = (f(x), b(x))$, where b is an *hardcore predicate* of f.[4] G's pseudoentropy is $n+1$, which is larger by one bit than its output (and input) entropy. Similar constructions can be applied to one-way functions that can be converted to (almost) injective one-way functions (e.g., regular one-way functions), but generalizing it to *arbitrary* one-way function is seemingly a much more challenging task. Yet, Håstad et al. [17] did manage to get a pseudoentropy generator out of an arbitrary one-way function, alas with poor parameters compared with what can easily be achieved from an injective one-way function. Specifically, while its output pseudoentropy is larger than its real output entropy, and thus it possesses a positive entropy gap, its entropy gap is tiny (i.e., $\log n/n$), and its pseudoentropy is smaller than its input length. In addition, the quantity of its pseudoentropy is not efficiently computable. These issues result in a complicated and indirect PRG construction. Constructions that followed this approach ([13, 19]), while improving and simplifying the original construction, also ended up being rather complicated and inefficient. To deal with this barrier, Haitner, Reingold, and Vadhan [14] presented a relaxation of this notion called *next-block pseudoentropy*, which can be easily obtained with strong parameters from any one-way function, yet is still strong enough for construction of PRGs.

[1] I.e., $|\Pr[\mathsf{D}(X) = 1] = \Pr[\mathsf{D}(U_n) = 1]| = \operatorname{neg}(n)$ for any polynomial-time distinguisher D, where U_n is uniformly distributed over $\{0,1\}^n$, and $\operatorname{neg}(n)$ is smaller than any inverse polynomial. See Section 2 for the formal definitions.

[2] The *Shanon entropy* of a random variable X is defined by $\mathrm{H}(X) = \mathrm{E}_{x \leftarrow X}\left[\log \frac{1}{\Pr[X=x]}\right]$.

[3] Håstad et al. [17] refer to such a generator as a *false entropy generator*, and require a pseudoentropy generator to have output pseudoentropy (at least) $n + \Delta(n)$, rather than just $\mathrm{H}(G(U_n)) + \Delta(n)$. For the sake of this exposition, however, we ignore this distinction.

[4] b is hardcore predicate of f if $(f(U_n), b(U_n))$ is computationally indistinguishable from $(f(U_n), U)$, for U_n and U sampled, uniformly and independently, from $\{0,1\}^n$ and $\{0,1\}$, respectively.

1.1.1 Next-Block Pseudoentropy

Next-block pseudoentropy is similar in spirit to the Blum and Micali [3] notion of next-bit unpredictability, which was shown by Yao [32] to be equivalent to his (now-standard) definition of pseudorandomness. This equivalence says that a random variable $X = (X_1, \ldots, X_m)$ is pseudorandom iff each bit of X is *unpredictable* from the previous bits. That is, $\Pr\left[\mathsf{P}(X_1, X_2, \ldots, X_{i-1}) = X_i\right] \le \frac{1}{2} + \operatorname{neg}(n)$ for every i and efficient predictor (i.e., algorithm) P. Equivalently, $(X_1, X_2, \ldots, X_{i-1}, X_i)$ is computationally indistinguishable from $(X_1, X_2, \ldots, X_{i-1}, U)$ where U is a uniform bit. It is thus natural to consider what happens if we relax the pseudorandomness of X_i to *pseudoentropy* (capturing the idea that X_i is only somewhat unpredictable from the previous bits). And more generally, we can allow the X_i's to be blocks instead of bits.

Definition 2 (next-block pseudoentropy [14], informal). *A random variable $X = (X_1, \ldots, X_m)$ is said to have* next-block pseudoentropy (at least) k *if there exists a sequence of random variables $Y = (Y_1, \ldots, Y_m)$, jointly distributed with X, such that:*

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

A function $G : \{0,1\}^n \mapsto (\{0,1\}^\ell)^m$ is said to have next-block pseudoentropy k *if $(X_1, \ldots, X_m) = G(U_n)$ has next-block pseudoentropy k. A* next-block pseudoentropy generator *is a polynomial-time computable function $G : \{0,1\}^n \mapsto (\{0,1\}^\ell)^m$ that has next-block pseudoentropy (at least) $\mathrm{H}(G(U_n)) + \Delta(n)$ for some $\Delta(n) > 1/\operatorname{poly}(n)$, where again Δ is called the* entropy gap.

That is, in total, the blocks of X "look like" they have k bits of entropy given the previous ones. Note that the case $k = m$ and blocks of size one (the X_i's are bits) amounts to the Yao [32] definition of unpredictability discussed above. The case of one block ($m = 1$) amounts to Håstad et al. [17] definition of pseudoentropy (Definition 1). 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.

Unlike the case of (HILL) pseudoentropy, it is known how to use any one-way function to construct a next-block pseudoentropy generator with good parameters.

Constructing next-block pseudoentropy generators from one-way functions. Given a one-way function $f\colon \{0,1\}^n \mapsto \{0,1\}^n$, we construct a generator G as

$$G(x) = (f(x), x_1, \ldots, x_n). \tag{.1}$$

The above construction was proven to achieve next-block pseudoentropy by Vadhan and Zheng [29]. The original construction of Haitner et al. [14] considered instead $G(x,h) = (f(x), h(x)_1, \ldots, h(x)_n)$, for an appropriate family of hash functions with seed length $O(n)$. In this tutorial, we will analyze the latter construction, using a family of hash functions of seed length $O(n^2)$, as it has a simpler analysis.[5]

If we consider only the original notion of pseudoentropy (Definition 1), the above construction is problematic; the polynomial-time test $T(y,x)$ that checks whether $y = f(x)$, distinguishes $G(U_n)$ from every random variable of entropy noticeably larger than n (since T accepts only 2^n strings). However, it turns out that it does have *next-block pseudoentropy at least* $n + \log n$. This has two advantages compared with the pseudoentropy generator constructed by Håstad et al. [17]. First, the entropy gap is now $\Delta = \log n$ instead of $\Delta = \log n/n$. Second, the total amount of pseudoentropy in the output (though not the amount contributed by the individual blocks) is known. These two advantages together yield a simpler and more efficient one-way function based PRG.

[5] Interestingly, the construction we consider in this tutorial is similar to the pseudoentropy generator used by Håstad et al. [17], but here it is viewed as a next-block pseudoentropy generator.

1.2 Inaccessible Entropy

Notions of pseudoentropy as above are only useful as a *lower* bound on the "computational entropy" in a distribution. For instance, it can be shown that *every* distribution on $\{0,1\}^n$ is computationally indistinguishable from a distribution of entropy at most $\operatorname{polylog} n$. In this section we introduce another computational analogue of entropy, which we call *accessible entropy*, which is useful as an *upper* bound on computational entropy. We motivate the idea of accessible entropy with an example. Let G be the following two-block generator:

Algorithm 1 (G)**.**
Let $m \ll n$ and let $\mathcal{H} = \{h\colon \{0,1\}^n \mapsto \{0,1\}^m\}$ be a family of collision-resistant hash functions.[6]

On public parameter $h \overset{R}{\leftarrow} \mathcal{H}$.

```
1. Sample $x \overset{R}{\leftarrow} \{0,1\}^n$.
2. Output $y = h(x)$.
3. Output $x$.
```

Now, information-theoretically, G's second output block (namely x) has entropy at least $n - m \geq 1$ conditioned on h and its first output block y. This is since $(h, y = h(x))$ reveals only m bits of information about x. The collision-resistance property of h, however, implies that given the *state* of G after it outputs its first block y, there is at most one consistent value of x that can be computed in polynomial time with nonnegligible probability. (Otherwise, we would be able find two distinct messages $x \neq x'$ such that $h(x) = h(x')$.) This holds even if G is replaced by any polynomial-time cheating strategy $\widetilde{G}$. Thus, there is "real entropy" in x (conditioned on h and the first output of G), but it is "computationally inaccessible" to $\widetilde{G}$, to whom x effectively has entropy 0.

We generalize this basic idea to allow the upper bound on the "accessible entropy" to be a parameter k, and to consider both the real and accessible entropy accumulated over several blocks of a generator. In more detail, consider an *m-block generator* $G\colon \{0,1\}^n \mapsto (\{0,1\}^*)^m$, and let $(Y_1, \ldots, Y_m)$ be random variables denoting the m output blocks generated by applying G over randomness U_n (no public parameters are given). We define the *real entropy* of G as $\mathrm{H}(G(U_n))$, the Shannon entropy of $G(U_n)$, which is equal to

$$\sum_{i \in [m]} \mathrm{H}(Y_i \mid Y_1, \ldots, Y_{i-1}),$$

where $\mathrm{H}(X \mid Y) = \mathrm{E}_{y \overset{R}{\leftarrow} Y}[\mathrm{H}(X|_{Y=y})]$ is the standard notion of (Shannon) conditional entropy.

To define *accessible entropy*, consider a probabilistic polynomial-time cheating strategy $\widetilde{G}$ that before outputting the i-th block, tosses some fresh random coins r_i, and uses them to output a string y_i. We restrict out attention to *G-consistent* (adversarial) generators—$\widetilde{G}$'s output is always in the support of G (though it might be distributed differently). Now, let $(R_1, Y_1, \ldots, Y_m, R_m)$ be random variables corresponding to a random execution of $\widetilde{G}$. We define the *accessible entropy* achieved by $\widetilde{G}$ to be

$$\sum_{i \in [m]} \mathrm{H}(Y_i \mid R_1, \ldots, R_{i-1}).$$

The key point is that now we compute the entropy conditioned not just on the previous output blocks $Y_1, \ldots, Y_{i-1}$ (which are determined by $R_1, \ldots, R_{i-1}$), as done when computing the real entropy of G, but also on the local state of $\widetilde{G}$ prior to outputting the i-th block (which without loss of generality equal its coin tosses $R_1, \ldots, R_{i-1}$). We define the *accessible entropy* of G as the maximal accessible entropy achieved by a G-consistent, polynomial-time generator $\widetilde{G}$. We refer to the difference (real entropy) $-$ (accessible entropy)

[6] Given $h \overset{R}{\leftarrow} \mathcal{H}$, it is infeasible to find distinct $x, x' \in \{0,1\}^n$ with $h(x) = h(x')$.

as the *inaccessible entropy* of the generator G, and call G an *inaccessible entropy generator* if its inaccessible entropy is noticeably greater than zero.

It is important to note that if we put no computational restrictions on the computational power of a G-consistent $\widetilde{G}$, then its accessible entropy can always be as high as the real entropy of G; to generate its i-th block y_i, $\widetilde{G}$ samples x uniformly at random from the set $\{x' \colon \mathsf{G}(x')_1 = y_1, \ldots, \mathsf{G}(x')_{i-1} = y_{i-1}\}$. This strategy, however, is not always possible for a computationally bounded $\widetilde{G}$.

The collision resistance example given earlier provides evidence that when allowing public parameters, there are efficient generators whose computationally accessible entropy is much smaller than their real Shannon entropy. Indeed, the real entropy of the generator we considered above is n (namely, the total entropy in x), but its accessible entropy is at most $m+\text{neg}(n) \ll n$, where m is the output length of the collision-resistant hash function.

As we shall see, we do not need collision resistance; any one-way function can be used to construct an inaccessible entropy generator (without public parameters). An application of this result is an alternative construction of statistically hiding commitment schemes from arbitrary one-way functions. This construction is significantly simpler and more efficient than the previous construction of Haitner et al. [10]. It also conceptually unifies the construction of statistically hiding commitments from one-way functions with the construction of pseudorandom generators discussed in the previous section: the first step of both constructions is to show that the one-way function directly yields a generator with a gap between its real entropy and "computational entropy" (pseudoentropy in the case of pseudorandom generators, and accessible entropy in the case of statistically hiding commitments). This gap is then amplified by repetitions and finally combined with various forms of hashing to obtain the desired primitive.

Constructing an inaccessible entropy generator from one-way functions. For a one-way function $f \colon \{0,1\}^n \mapsto \{0,1\}^n$, consider the $(n+1)$-block generator

$$G(x) = (f(x)_1, f(x)_2, \ldots, f(x)_n, x).$$

Notice that this construction is the same as the construction of a next-block pseudoentropy generator from a one-way function (Construction .1), except that we have broken $f(x)$ into one-bit blocks rather than breaking x. Again, the real entropy of $G(U_n)$ is n. It can be shown that the accessible entropy of G is at most $n - \log n$, so again we have an entropy gap of $\log n$ bit.

1.3 Rest of This Tutorial

Standard notations, definitions, and facts, are given in Section 2. An elaborated discussion of next-block pseudoentropy, containing formal definitions, a construction from one-way functions, and its use in constricting pseudorandom generators, is given in Section 3. An elaborated discussion of inaccessible entropy, with formal definitions, a construction from one-way functions, and its use in constructing statistically hiding commitment schemes, is given in Section 4. In both sections, we have chosen simplicity and clarity over full generality and efficiency. For details of the latter, see the Further Reading section below.

1.4 Related Work and Further Reading

Pseudoentropy. More details and improvements on the construction of pseudorandom generator from one-way functions via next-block pseudoentropy can be found in the works of Haitner et al. [14] and Vadhan and Zheng [29]. In particular, Vadhan and Zheng [29] also show how to save a factor of n in the seed-length blow up in the reduction from next-block pseudoentropy generator to PRG, thereby reducing the seed length from $\tilde{O}(n^4)$ to $\tilde{O}(n^3)$ (at the price of making adaptive calls to the one-way function). Holenstein and Sinha [20] showed that any black-box construction of a pseudorandom generator from a one-way function on n-bit inputs must invoke the one-way function $\Omega(n/\log n)$ times. Their lower bound also applies to regular one-way functions (of unknown regularity), and is tight in this case (due to the constructions of [8, 13]). The constructions of Haitner et al. [14] and of Vadhan and Zheng [29] from arbitrary one-way functions invoke the one-way function $\tilde{O}(n^3)$ times. It remains open whether the super linear number of invocations or the super-linear seed length is necessary, or the constructions can be furthered improved.

Several other computational analogues of entropy have been studied in the literature (cf. [1, 21]), all of which serve as ways of capturing the idea that a distribution "behaves like" one of higher entropy.

Inaccessible entropy. The details of the construction of statistically hiding commitments from one-way functions via inaccessible entropy can be found in the work of Haitner et al. [16]. A preliminary version of that paper [11] uses a more general, and more complicated, notion of accessible entropy which measures the accessible entropy of *protocols* rather than generators. This latter notion is used in [11] to show that, if NP has constant-round interactive proofs that are black-box zero knowledge under parallel composition, then there exist constant-round statistically hiding commitment schemes. A subsequent work of Haitner et al. [12] uses a simplified version of accessible entropy to present a simpler and more efficient construction of *universal one-way functions* from any one-way function. One of the two inaccessible entropy generators considered in [12], for constructing universal one-way functions, is very similar to the constructionist next-block pseudoentropy and inaccessible entropy generators discussed above (in Sections 1.1 and 1.2). Hence, all of these three notions of computational entropy can be found in any one-way function using very similar constructions, all simple variants of $G(x) = (f(x), x)$, where f is an arbitrary one-way function.

The notion of inaccessible entropy, of the simpler variant appearing in [12], is in a sense implicit in the work of Rompel [27], who first showed how to base universal one-way functions on any one-way functions.

2 Preliminaries

2.1 Notation

We use calligraphic letters to denote sets, upper-case for random variables, lower-case for values, bold-face for vectors. and sanserif for algorithms (i.e., Turing machines). For $n \in \mathbb{N}$, let $[n] = \{1, \ldots, n\}$. For vector $\mathbf{y} = (y_1, \ldots, y_n)$ and $\mathcal{J} \subseteq [n]$, let $\mathbf{y}_{\mathcal{J}} = (y_{i_1}, \ldots, y_{i_{|\mathcal{J}|}})$, where $i_1 < \ldots < i_{|\mathcal{J}|}$ are the elements of $\mathcal{J}$. Let $\mathbf{y}_{<j} = \mathbf{y}_{[j-1]} = (y_1, \ldots, y_{j-1})$ and $\mathbf{y}_{\leq j} = \mathbf{y}_{[j]} = (y_1, \ldots, y_j)$. Both notations naturally extend to an ordered list of elements that is embedded in a larger vector (i.e., given $(a_1, b_1, \ldots, a_n, b_n)$, $a_{<3}$ refers to the vector (a_1, a_2)). Let poly denote the set of all positive polynomials, let $\mathsf{ppt}^{\mathsf{NU}}$ stand for a *nonuniform* probabilistic polynomial-time algorithm. A function $\nu \colon \mathbb{N} \mapsto [0, 1]$ is *negligible*, denoted $\nu(n) = \operatorname{neg}(n)$, if $\nu(n) < 1/p(n)$ for every $p \in \operatorname{poly}$ and large enough n. For a function f and a set $\mathcal{S}$, let $\operatorname{Im}(f(\mathcal{S})) = \{f(x) \colon x \in \mathcal{S}\}$.

2.2 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. For an event E, we write $X|_E$ to denote the random variable X conditioned on E. We let $\Pr_{X|Y}[x|y]$ stand for $\Pr[X = x \mid Y = y]$. The *support* of a random variable X, denoted $\operatorname{Supp}(X)$, is defined as $\{x \colon \Pr[X = x] > 0\}$. The variable X is *flat* if it is uniform on its support. Let U_n denote a random variable that is uniform over $\{0, 1\}^n$. For $t \in \mathbb{N}$, let $X^{(t)} = (X^1, \ldots, X^t)$, where $X^1, \ldots, X^t$ are independent copies of X.

We write $X \equiv Y$ to indicate that X and Y are identically distributed. We write $\Delta(X,Y)$ to denote the *statistical difference* (also known as 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 $\Delta(X,Y) \leq \varepsilon$ [resp., $\Delta(X,Y) > \varepsilon$], we say that X and Y are *ε-close* [resp., ε-far]. Two random variables $X = X(n)$ and $Y = Y(n)$ are *statistically indistinguishable*, denoted $X \approx_{\mathsf{S}} Y$, if for any unbounded algorithm D, it holds that $|\Pr[\mathsf{D}(1^n, X(n)) = 1] - \Pr[\mathsf{D}(1^n, Y(n)) = 1]| = \operatorname{neg}(n)$.[7] Similarly, X and Y are *nonuniformly* computationally indistinguishable, denoted $X \approx_{\mathsf{nu\text{-}C}} Y$], if $|\Pr[\mathsf{D}(1^n, X(n)) = 1] - \Pr[\mathsf{D}(1^n, Y(n)) = 1]| = \operatorname{neg}(n)$ for every $\mathsf{ppt}^{\mathsf{NU}}$ D.

2.3 Entropy Measures

We refer to several measures of entropy. 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 $x \in \operatorname{Supp}(X)$, the *sample-entropy* of x with respect to X is the quantity

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

letting $\mathrm{H}_X(x) = \infty$ for $x \notin \operatorname{Supp}(X)$, and $2^{-\infty} = 0$.

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:

$$\mathrm{H}(X) := \mathop{\mathrm{E}}_{x \overset{\mathrm{R}}{\leftarrow} X} [\mathrm{H}_X(x)],$$
$$\mathrm{H}_\infty(X) := \min_{x \in \operatorname{Supp}(X)} \mathrm{H}_X(x).$$

The *collision probability* of X is defined by $\mathrm{CP}(X) := \sum_{x \in \operatorname{Supp}(X)} \Pr_X[x]^2 = \Pr_{(x,x') \overset{\mathrm{R}}{\leftarrow} X^2}[x = x']$, and its *Rényi-entropy* is defined by

$$\mathrm{H}_2(X) := -\log \mathrm{CP}(X).$$

We will also discuss the *max-entropy* $\mathrm{H}_0(X) := \log(1/|\operatorname{Supp}(X)|)$. The term "max-entropy" and its relation to the sample-entropy will be made apparent below.

It can be shown that $\mathrm{H}_\infty(X) \leq \mathrm{H}_2(X) \leq \mathrm{H}(X) \leq \mathrm{H}_0(X)$ with each inequality being an equality if and only if X is flat. Thus, saying $\mathrm{H}_\infty(X) \geq k$ is a strong way of saying that X has "high entropy" and $\mathrm{H}_0(X) \leq k$ a strong way of saying that X as "low entropy".

The following fact quantifies the probability that the sample-entrose is larger than the max-entropy.

Lemma 1. *For random variable X it holds that*

1. $\mathrm{E}_{x \overset{R}{\leftarrow} X}\left[2^{\mathrm{H}_X(x)}\right] = |\operatorname{Supp}(X)|$.
2. $\Pr_{x \overset{R}{\leftarrow} X}\left[\mathrm{H}_X(x) > \log \frac{1}{\varepsilon} + \mathrm{H}_0(X)\right] < \varepsilon$, *for any* $\varepsilon > 0$.

Proof. For the first item, compute

[7] This is equivalent to asking that $\Delta(X(n), Y(n)) = \operatorname{neg}(n)$.

$$\begin{aligned}\mathop{\mathrm{E}}_{x \overset{\mathrm{R}}{\leftarrow} X}\left[2^{\mathrm{H}_X(x)}\right] &= \sum_{x \in \mathrm{Supp}(X)} 2^{-\mathrm{H}_X(x)} \cdot 2^{\mathrm{H}_X(x)} \\ &= \sum_{x \in \mathrm{Supp}(X)} 1 \\ &= |\mathrm{Supp}(X)| .\end{aligned}$$

The second item follows by the first item and Markov inequality.

$$\begin{aligned}\Pr_{x \overset{\mathrm{R}}{\leftarrow} X}\left[\mathrm{H}_X(x) > \log\frac{1}{\varepsilon} + \mathrm{H}_0(X)\right] &= \Pr_{x \overset{\mathrm{R}}{\leftarrow} X}\left[2^{\mathrm{H}_X(x)} > \frac{1}{\varepsilon} \cdot |\mathrm{Supp}(X)|\right] \\ &< \varepsilon.\end{aligned}$$

□

Conditional entropies. We will also be interested in conditional versions of entropy. For jointly distributed random variables (X, Y) and $(x, y) \in \mathrm{Supp}(X, Y)$, we define the *conditional sample-entropy* to be $\mathrm{H}_{X|Y}(x|y) = \log \frac{1}{\Pr_{X|Y}[x|y]} = \log \frac{1}{\Pr[X=x|Y=y]}$. Then the standard *conditional Shannon entropy* can be written as

$$\mathrm{H}(X \mid Y) = \mathop{\mathrm{E}}_{(x,y) \overset{\mathrm{R}}{\leftarrow} (X,Y)}\left[\mathrm{H}_{X|Y}(x \mid y)\right] = \mathop{\mathrm{E}}_{y \overset{\mathrm{R}}{\leftarrow} Y}\left[\mathrm{H}(X|_{Y=y})\right] = \mathrm{H}(X, Y) - \mathrm{H}(Y).$$

The following known lemma states that conditioning on a "short" variable is unlikely to change the sample-entropy by much.

Lemma 2. *Let X and Y be random variables, let $k = \mathrm{H}_\infty(X)$, and let $\ell = \mathrm{H}_0(Y)$. Then, for any $t > 0$, it holds that*

$$\Pr_{(x,y) \overset{R}{\leftarrow} (X,Y)}\left[\mathrm{H}_{X|Y}(x|y) < k - \ell - t\right] < 2^{-t}.$$

Proof. For $y \in \mathrm{Supp}(Y)$, let $\mathcal{X}_y = \{x \in \mathrm{Supp}(X) \colon \mathrm{H}_{X|Y}(x|y) < k - \ell - t\}$. We have $|\mathcal{X}_y| < 2^{k-\ell-t}$. Hence, $\left|\mathcal{X} = \bigcup_{y \in \mathrm{Supp}(Y)} \mathcal{X}_y\right| < 2^\ell \cdot 2^{k-\ell-t} = 2^{k-t}$. It follows that

$$\Pr_{(x,y) \overset{\mathrm{R}}{\leftarrow} (X,Y)}\left[\mathrm{H}_{X|Y}(x|y) < k - \ell - t\right] \le \Pr_{(x,y) \overset{\mathrm{R}}{\leftarrow} (X,Y)}\left[x \in \mathcal{X}\right] < 2^{-k} \cdot 2^{k-t} = 2^{-t}.$$

□

Smoothed entropies. The following lemma will allow us to think of a random variable X whose sample-entropy is high with high probability as if it has high min-entropy (i.e., as if its sample-entropy function is "smoother", with no picks).

Lemma 3. *Let X, Y be random variable and let $\varepsilon > 0$.*

1. *Suppose $\Pr_{x \overset{R}{\leftarrow} X}\left[\mathrm{H}_X(x) \ge k\right] \ge 1 - \varepsilon$, then X is ε-close to a random variable X' with $\mathrm{H}_\infty(X') \ge k$.*
2. *Suppose $\Pr_{(x,y) \overset{R}{\leftarrow} (X,Y)}\left[\mathrm{H}_{X|Y}(x|y) \ge k\right] \ge 1 - \varepsilon$, then (X, Y) is ε-close to a random variable (X', Y') with $\mathrm{H}_{X'|Y'}(x|y) \ge k$ for any $(x, y) \in \mathrm{Supp}(X', Y')$. Further, Y' and Y are identically distributed.*

Proof Sketch. For the first item, we modify X on an ε fraction of the probability space (corresponding to when X takes on a value x such that $\mathrm{H}_X(x) \ge k$) to bring all probabilities to be smaller than or equal to 2^{-k}.

The second item is proved via similar means, while when changing (X, Y), we do so without changing the "Y" coordinate.

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 4 ([31], Theorem 3.14). *Let X be a random variables taking values in a universe $\mathcal{U}$, let $t \in \mathbb{N}$, and let $0 < \varepsilon \leq 1/e^2$. Then with probability at least $1-\varepsilon$ over $x \overset{R}{\leftarrow} X^{(t)}$,*

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

We will make use of the following "conditional variant" of Lemma 4:

Lemma 5. *Let X and Y be jointly distributed random variables where X takes values in a universe $\mathcal{U}$, let $t \in \mathbb{N}$, and let $0 < \varepsilon \leq 1/e^2$. Then with probability at least $1-\varepsilon$ over $(x,y) \leftarrow (X',Y') = (X,Y)^{(t)}$,*

$$\mathrm{H}_{X'|Y'}(x \mid y) - t \cdot \mathrm{H}(X \mid Y) \geq -O\left(\sqrt{t \cdot \log \tfrac{1}{\varepsilon}} \cdot \log(|\mathcal{U}| \cdot t)\right).$$

The proof of Lemma 5 follows the same line as the proof of Lemma 4, by considering the random variable $\mathrm{H}_{X|Y}(X|Y)$ instead of $\mathrm{H}_X(X)$.

Sub-additivity. The chain rule for Shannon entropy yields that

$$\mathrm{H}(X = (X_1, \ldots, X_t)) = \sum_i \mathrm{H}(X_i | X_1, \ldots, X_{i-1}) \leq \sum_i \mathrm{H}(X_i).$$

The following lemma shows that a variant of the above also holds for sample-entropy.

Lemma 6. *For random variables $\mathbf{X} = (X_1, \ldots, X_t)$, it holds that*

1. $\mathrm{E}_{\mathbf{x} \overset{R}{\leftarrow} \mathbf{X}}\left[2^{\mathrm{H}_{\mathbf{X}}(\mathbf{x}) - \sum_t \mathrm{H}_{X_i}(\mathbf{x}_i)}\right] \leq 1.$
2. $\Pr_{\mathbf{x} \overset{R}{\leftarrow} \mathbf{X}}\left[H_{\mathbf{X}}(\mathbf{x}) > \log \frac{1}{\varepsilon} + \sum_{i \in [t]} H_{X_i}(\mathbf{x}_i)\right] < \varepsilon$, *for any* $\varepsilon > 0$.

Proof. As in Lemma 1, the second part follows from the first by Markov's inequality. For the first part, compute

$$\begin{aligned}\mathop{\mathrm{E}}_{\mathbf{x} \overset{R}{\leftarrow} \mathbf{X}}\left[2^{\mathrm{H}_{\mathbf{X}}(\mathbf{x}) - \sum_t \mathrm{H}_{X_i}(\mathbf{x}_i)}\right] &= \sum_{\mathbf{x} \in \mathrm{Supp}(\mathbf{X})} \Pr[\mathbf{X} = \mathbf{x}] \cdot \frac{\prod_{i \in [t]} \Pr[X_i = \mathbf{x}_i]}{\Pr[\mathbf{X} = \mathbf{x}]} \\ &= \sum_{\mathbf{x} \in \mathrm{Supp}(\mathbf{X})} \prod_i \Pr[X_i = \mathbf{x}_i] \\ &\leq 1.\end{aligned}$$

□

The following lemma generalizes Lemma 1 to settings that come up naturally when upper bounding the accessible entropy of a generator (as we do in Section 4):

Definition 3. *For a t-tuple random variable $\mathbf{X} = (X_1, \ldots, X_t)$, $\mathbf{x} \in \mathrm{Supp}(X)$ and $\mathcal{J} \subseteq [t]$, let*

$$\mathrm{H}_{\mathbf{X},\mathcal{J}}(\mathbf{x}) = \sum_{i \in \mathcal{J}} \mathrm{H}_{\mathbf{X}_i | \mathbf{X}_{<i}}(\mathbf{x}_i | \mathbf{x}_{<i}).$$

Lemma 7. *Let $\mathbf{X} = (X_1, \ldots, X_t)$ be a sequence of random variables and let $\mathcal{J} \subseteq [t]$. Then,*

1. $\mathrm{E}_{\mathbf{x} \overset{R}{\leftarrow} \mathbf{X}}\left[2^{\mathrm{H}_{\mathbf{X},\mathcal{J}}(\mathbf{x})}\right] \leq |\mathrm{Supp}(X_{\mathcal{J}})|.$

2. $\Pr_{x \overset{R}{\leftarrow} X}\left[\mathrm{H}_{\mathbf{X},\mathcal{J}}(\mathbf{x}) > \log\frac{1}{\varepsilon} + \mathrm{H}_0(X_{\mathcal{J}})\right] < \varepsilon$*, for any* $\varepsilon > 0$.

Proof. The second item follows from the first one as in the proof of Lemma 1. We prove the first item by induction on t and $|J|$. The case $t = 1$ is immediate, so we assume for all $(t', \mathcal{J}')$ with $(t', |\mathcal{J}'|) < (t, |\mathcal{J}|)$ and prove it for $(t, \mathcal{J})$. Assume that $1 \in \mathcal{J}$ (the case $1 \notin \mathcal{J}$ is analogous) and let $\mathbf{X}_{-1} = (X_2, \ldots, X_t)$ and $\mathcal{J}_{-1} = \{i - 1 \colon i \in \mathcal{J} \setminus \{1\}\}$. Compute

$$\begin{aligned}
\mathop{\mathrm{E}}_{\mathbf{x} \overset{R}{\leftarrow} \mathbf{X}}\left[2^{\mathrm{H}_{\mathbf{X},\mathcal{J}}(\mathbf{x})}\right] &= \sum_{x_1 \in \mathrm{Supp}(X_1)} 2^{-\mathrm{H}_{X_1}(\mathbf{x}_1)} \cdot 2^{\mathrm{H}_{X_1}(x_1)} \cdot \mathop{\mathrm{E}}_{\mathbf{x} \overset{R}{\leftarrow} \mathbf{X}_{-1}|_{X_1 = x_1}}\left[2^{\mathrm{H}_{\mathbf{X}_{-1}|_{X_1=x_1}, \mathcal{J}_{-1}}(\mathbf{x})}\right] \\
&\leq \sum_{x_1 \in \mathrm{Supp}(X_1)} 1 \cdot \left|\mathrm{Supp}((\mathbf{X}_{-1})_{\mathcal{J}_{-1}}|_{X_1 = x_1})\right| \\
&= \sum_{x_1 \in \mathrm{Supp}(X_1)} \left|\mathrm{Supp}(\mathbf{X}_{\mathcal{J}\setminus\{1\}}|_{X_1 = x_1})\right| \\
&\leq \left|\mathrm{Supp}(\mathbf{X}_{\mathcal{J}})\right|.
\end{aligned}$$

□

2.4 Hashing

We will use two types of (combinatorial) "hash" functions.

2.4.1 Two-Universal Hashing

Definition 4 (Two-universal function family). *A function family* $\mathcal{H} = \{h \colon \mathcal{D} \mapsto \mathcal{R}\}$ *is* two universal *if* $\forall x \neq x' \in \mathcal{D}$*, it holds that* $\Pr_{h \leftarrow \mathcal{H}}[h(x) = h(x')] \leq 1/|\mathcal{R}|$.

An example of such a function family is the set $\mathcal{H}_{s,t} = \{0,1\}^{s \times t}$ of Boolean matrices, where for $h \in \mathcal{H}_{s,t}$ and $x \in \{0,1\}^s$, we let $h(x) = h \times x$ (i.e., the matrix vector product over GF_2). Another canonical example is $\mathcal{H}_{s,t} = \{0,1\}^s$ defined by $h(x) := h \cdot x$ over $\mathrm{GF}(2^s)$, truncated to its first t bits.

A useful application of two-universal hash functions is to convert a source of high Rényi entropy into a (close to) uniform distribution.

Lemma 8 (Leftover hash lemma [24, 23]). *Let* X *be a random variable over* $\{0,1\}^n$ *with* $\mathrm{H}_2(X) \geq k$*, let* $\mathcal{H} = \{g \colon \{0,1\}^n \mapsto \{0,1\}^m\}$ *be two-universal, and let* $H \overset{R}{\leftarrow} \mathcal{H}$*. Then* $\mathrm{SD}((H, H(X)), (H, \mathcal{U}_m)) \leq \frac{1}{2} \cdot 2^{(m-k)/2}$.

2.4.2 Many-wise Independent Hashing

Definition 5 (ℓ-Wise independent function family). *A function family* $\mathcal{H} = \{h \colon \mathcal{D} \mapsto \mathcal{R}\}$ *is* ℓ-wise independent *if for any distinct* $x_1, \ldots, x_\ell \in \mathcal{D}$*, it holds that* $(H(x_1), \ldots, H(x_\ell))$ *for* $H \overset{R}{\leftarrow} \mathcal{H}$ *is uniform over* $\mathcal{R}^\ell$.

The canonical example of such an ℓ-wise independent function family is $\mathcal{H}_{s,t,\ell} = (\{0,1\}^s)^\ell$ defined by $(h_0, \ldots, h_{\ell-i})(x) := \sum_{0 \leq i \leq \ell-1} h_i \cdot x^i$ over $\mathrm{GF}(2^s)$, truncated to its first t bits.

It is easy to see that, for $\ell > 1$, an ℓ-wise independent function family is two-universal, but ℓ-wise independent function families, in particular with larger value of ℓ, have stronger guarantees on their output distribution compared with two-universal hashing. We will state, and use, one such guarantee in the construction of statistically hiding commitment schemes presented in Section 4.

2.5 One-Way Functions

We recall the standard definition of one-way functions.

Definition 6 (one-way functions). *A polynomial-time computable $f\colon \{0,1\}^n \mapsto \{0,1\}^*$ is nonuniformly* one way *if for every* $\mathsf{ppt}^{\mathsf{NU}}$ A

$$\Pr_{y \leftarrow f(U_{s(n)})}\left[\mathsf{A}(1^n, y) \in f^{-1}(y)\right] = \operatorname{neg}(n). \tag{.2}$$

Without loss of generality, cf. [13], it can be assumed that $s(n) = n$ and f is length-preserving (i.e., $|f(x)| = |x|$).

3 Next-Block Entropy and Pseudorandom Generators

In this section, we formally define the notion of next-block pseudoentropy, and use it as intermediate tool to construct pseudorandom generators from one-way functions. Preferring clarity over generality, we present a simplified version of the definitions and constructions. For the full details see [14].

We start in Section 3.1, by presenting the formal definition of next-block pseudoentropy. In Section 3.2 we show that any one-way function can be used to construct a generator with a useful amount of next-block pseudoentropy. In Section 3.3 we develop means to manipulate next-block pseudoentropy. Finally, in Section 3.4, we show how to convert generators of the type constructed in Section 3.2 into pseudorandom generators, thus reproving the fundamental result that pseudorandom generators can be based on any one-way function.

3.1 Next-Block Pseudoentropy

Recall from the introduction that the next-block pseudoentropy is of a similar spirit to the Blum and Micali [3] notion of next-bit unpredictability; a random variable $X = (X_1, \ldots, X_m)$ is *next-bit unpredictable* if the bit X_i cannot be predicted with nonnegligible advantage from $X_{<i} = (X_1, X_2, \ldots, X_{i-1})$, or alternatively, X_i is pseudorandom given $X_{<i}$. Next-block pseudoentropy relaxes this notion by only requiring that X_i has some pseudoentropy given $X_{<i}$.

We now formally define the notion of next-block pseudoentropy for the cases of both Shannon entropy and min-entropy. The definition below differs from the definition of [14], in that we require the indistinguishability to hold (also) against *nonuniform adversaries*. This change simplifies the definitions and proofs (see Remark 1), but at the price that we can only construct such pseudoentropy generators from functions that are nonuniformly one-way (i.e., ones that are hard to invert for such nonuniform adversaries). We start by recalling the more standard definitions of pseudoentropy and pseudorandomness (to be consistent with the next-block pseudoentropy definitions given below, we give the nonuniform version of these definitions).

Definition 7 (Pseudoentropy and pseudorandomness). *Let n be a security parameter and $X = X(n)$ be a random variable distributed over strings of length $\ell(n) \leq \operatorname{poly}(n)$. We say that $X = X(n)$ has* pseudoentropy (at least) *$k = k(n)$ if there exists a random variable $Y = Y(n)$, such that*

1. *$\mathrm{H}(Y) \geq k$, and*
2. *X and Y are nonuniformly computationally indistinguishable. I.e., for every* $\mathsf{ppt}^{\mathsf{NU}}$ D*, it holds that*

$$\Pr\left[\mathsf{D}(1^n, X) = 1\right] - \Pr\left[\mathsf{D}(1^n, Y) = 1\right] = \operatorname{neg}(n).$$

If $\mathrm{H}_\infty(Y) \geq k$*, we say that* X *has* pseudo-min-entropy (at least) k*, where if* $k = \ell(n)$*, we say that* X *is* pseudorandom *(which is equivalent to asking that* X *is computationally nonuniformly indistinguishable from* U_ℓ*).*

Finally, a polynomial-time computable function $G\colon \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}$ *is a* pseudorandom generator *if* $\ell > n$ *and* $G(U_n)$ *is pseudorandom.*

That is, pseudoentropy is the computational analog of entropy. In construct, next-block pseudoentropy is a computational analog of unpredictability.

Definition 8. *(Next-block pseudoentropy) Let* $m = m(n)$ *be an integer function. A random variable* $X = X(n) = (X_1, \ldots, X_m)$ *is said to have* next-block (Shannon) pseudoentropy (at least) $k = k(n)$ *if there exists a (jointly distributed) random variable* $Y = Y(n) = (Y_1, \ldots, Y_m)$ *such that*

1. $\sum_{i=1}^{m} \mathrm{H}(Y_i \mid X_{<i}) \geq k$*, and*
2. Y *is* block-wise indistinguishable from X*: for every* ppt$^{\mathsf{NU}}$ D *and* $i = i(n) \in [m(n)]$*,*

$$\Pr\left[\mathsf{D}(1^n, X_{\leq i}) = 1\right] - \Pr\left[\mathsf{D}(1^n, X_{<i}, Y_i) = 1\right] = \mathrm{neg}(n).$$

Every block of X has next-block (Shannon) pseudoentropy *at least* $\alpha = \alpha(n)$ *if condition* 1 *above is replaced with*

1. $\mathrm{H}(Y_i|_{X_{<i}=x_{<i}}) \geq \alpha$*, for every* $x \in \mathrm{Supp}(X)$ *and* $i \in [m]$*.*

Every block of X has next-block pseudo-min-entropy *at least* α *if condition* 1 *above is replaced with*

1. $\mathrm{H}_\infty(Y_i|_{X_{<i}=x_{<i}}) \geq \alpha$*, for every* $x \in \mathrm{Supp}(X)$ *and* $i \in [m]$*.*

Finally, a generator G *over* $\{0,1\}^*$ *has* next-block pseudoentropy at least k *if (the random variable)* $G(U_n)$ *has. Similarly, every block of* G *has* next-block pseudoentropy [resp., pseudo-min-entropy] at least α *if* $G(U_n)$ *has.*

The above definitions naturally extend to generators that are only defined over some input lengths (e.g., on inputs of length $n^2 + n$ for all $n \in \mathbb{N}$). Our constructions directly yield such input-restricted generators, but since the inputs on which they are defined are the image of a polynomial (such as $n^2 + n$), they can be converted to ones defined on all inputs in a standard way.[8]

Throughout, we often omit the parameter n when its value is clear from the context.

Remark 1 (Uniform distinguishers). When working with a random variable X with a certain guarantee about its pseudoentropy (here as a generic name for the different types of pseudoentropy), one often likes to lower-bound the amount of pseudoentropy several independent copies of X have (jointly). Such lower bounds are used, for instance, in all constructions of pseudorandom generators from one-way functions [13, 14, 17, 19, 29]. Proving such lower bounds, however, typically requires the ability to sample efficiently from X, and also from a random variable Y that realizes the pseudoentropy of X (cf. Definition 7). While the X's in consideration are typically efficiently samplable, this is often not the case with respect to to the Y's. Considering nonuniform distinguishers bypasses this issue; such distinguishers can get the samples as a nonuniform advice. An alternative approach is to alter the definition of pseudoentropy to require that the random variables in consideration (i.e., X and Y) are computationally indistinguishable by (uniform) algorithms that have access to an oracle that samples from the joint distribution of (X, Y). This is the approach taken in [14], where the construction we present here is proven to be secure in uniform settings (in order to construct pseudorandom generators secure against uniform distinguishers, from one-way functions secure against uniform inverters.

[8] I.e., on input of arbitrary length, apply the input-restricted generator on the longest prefix of the input that matches the restricted set of lengths, and append the unused suffix of the input to the output.

3.2 Next-Block Pseudoentropy Generators from One-Way Functions

In this section, we show how to construct a next-block pseudoentropy generator G^f_{nb} out of a one-way function $f\colon \{0,1\}^n \mapsto \{0,1\}^n$.

Notation 2. *For $n, \ell \in \mathbb{N}$, let $\mathcal{H}_{n,\ell}$ be the family of $\ell \times n$ Boolean matrices, and let $\mathcal{H}_n = \mathcal{H}_{n,n}$. For $h \in \mathcal{H}_{n,\ell}$ and $x \in \{0,1\}^n$, let $h(x) = hx$ (i.e., the matrix vector product over $\mathrm{GF}(2)$). Throughout, we denote by H_n the random variable that is uniformly distributed over $\mathcal{H}_n$.*

Definition 9. *On $x \in \{0,1\}^n$, $h \in \mathcal{H}_n$, and $f\colon \{0,1\}^n \mapsto \{0,1\}^n$, define $G^f_{\mathrm{nb}}\colon \{0,1\}^n \times \mathcal{H}_n \mapsto \{0,1\}^n \times \mathcal{H}_n \times \{0,1\}^n$ by*

$$G^f_{\mathrm{nb}}(x,h) = (f(x), h, h(x)).$$

Theorem 3. *Let $f\colon \{0,1\}^n \mapsto \{0,1\}^n$ and let $G_{\mathrm{nb}} = G^f_{\mathrm{nb}}$ be according to Definition 9, viewed as a $(t(n) = n^2 + 2n)$-block generator (i.e., each output bit forms a separate block) over $(s(n) = n^2 + n)$-bit strings. Assuming f is nonuniformly one-way, then G^f_{nb} has next-block pseudoentropy at least $s(n) + c \cdot \log n$, for any $c > 0$.*

Namely, the next-block pseudoentropy of G^f_{nb} is $\log n$ bits larger than its input entropy.

Remark 2 (Tighter reductions). Haitner et al. [14] proved a variant of Theorem 3 in which the family H_n is replaced by a more sophisticated function family of description length $\Theta(n)$. As discussed in the introduction, Vadhan and Zheng [29] took this a step further and proved a variant of this theorem without using any function family. That is, they proved that $G^f_{\mathrm{nb}}(x) = (f(x), x)$ has next-block pseudoentropy $n + \log n$. In both cases, the gap between the real entropy of the output and the next-block pseudoentropy is $\log n$, as in Theorem 3, but the input length is only $\Theta(n)$ (versus $\Theta(n^2)$ in Theorem 3). This better ratio between the entropy gap and the input length yields a final pseudorandom generator of much shorter seed length (see Theorem 7). Both constructions, and in particular that of [29], require a more sophisticated analysis than the one we present here (also in their nonuniform forms).

A key step towards proving Theorem 3 is analyzing the following (possibly inefficient) function g^f:

Definition 10. *For $f\colon \{0,1\}^n \mapsto \{0,1\}^n$, let $\mathsf{D_f}(y) = \left\lceil \log \left| f^{-1}(y) = \{x \in \{0,1\}^n \colon f(x) = y\} \right| \right\rceil$, and define g^f over $\{0,1\}^n \times \mathcal{H}_n$, by*

$$g^f(x,h) = (f(x), h, h(x)_{1,\ldots,\mathsf{D_f}(f(x))}).$$

That is, $g(x,h)$ outputs a prefix of $G_{\mathrm{nb}}(x,h)$ whose length depends on the "degeneracy" of $f(x)$. What makes g interesting is that it is both close to being injective and hard to invert. To see this, note that $\mathrm{H}_\infty(U_n|_{f(U_n)=y}) = \mathrm{H}_0(U_n|_{f(U_n)=y}) = \log \left| f^{-1}(y) \right| \approx \mathsf{D_f}(y)$. Hence, the two-universality of $\mathcal{H}$ implies that $g(U_n, H_n)$ *determines* U_n with constant probability. In other words, $g(U_n, H_n)$ has a single preimage with constant probability. But the two-universality of $\mathcal{H}$ also yields that, for every $k(U_n) = \mathsf{D_f}(f(U_n)) - \omega(\log n)$, it holds that $H_n(U_n)_{1,\ldots,k(U_n)}$ is statistically *close to uniform* given $(f(U_n), H_n)$. Hence, $H_n(U_n)_{1,\ldots,\mathsf{D_f}(f(U_n))}$ does not provide enough information to enable an efficient inversion of f. (The extra $O(\log n)$ bits beyond $k(U_n)$ can only increase the inversion probability by a $\mathrm{poly}(n)$ factor.)

The following claims state formally the two properties of g mentioned above. The first claim states that the collision probability of g is small,[9] yielding that g has high entropy.

Claim 4. *Let $f\colon \{0,1\}^n \mapsto \{0,1\}^n$ and let $g = g^f$ as in Definition 10. Then $\mathrm{CP}(g(U_n, H_n)) \leq \frac{3}{|\mathcal{H}_n \times \{0,1\}^n|}$.*

Definition 11 (Hard-to-invert functions). *A function $q\colon \{0,1\}^n \mapsto \{0,1\}^*$ is* (nonuniformly) hard to invert *if $\Pr_{y \xleftarrow{R} q(U_n)} \left[\mathsf{A}(1^n, y) \in q^{-1}(y) \right] = \mathrm{neg}(n)$ for every $\mathsf{ppt}^{\mathsf{NU}}$ A.*

[9] Recall that the collision probability of a random variable X is defined as $\mathrm{CP}(X) = \Pr_{(x,x') \xleftarrow{R} X^2} [x = x']$, and that its Rényi entropy defined by $\mathrm{H}_2(X) = -\log \mathrm{CP}(X)$ lower-bounds its Shannon entropy.

Namely, an hard-to-invert function is a one-way function without the efficient computability requirement.

Claim 5. *Let $f\colon \{0,1\}^n \mapsto \{0,1\}^n$ and let $g = g^f$ be according to Definition 10. Assuming f is nonuniformly one-way, g is hard to invert.*

The proof of the above claims is given below, but first let us use it for proving Theorem 3. We will also use the Goldreich-Levin hardcore lemma.

Lemma 9 (Goldreich-Levin hardcore lemma, [7]). *Let $q\colon \{0,1\}^n \mapsto \{0,1\}^*$ be a hard-to-invert function and let $\ell = \ell(n) \in O(\log n)$, then $(q(U_n), H_{n,\ell}, H_{n,\ell}(U_n))$ is nonuniform computationally indistinguishable from $(q(U_n), H_{n,\ell}, U'_\ell)$.*[10]

Proving Theorem 3.

Proof of Theorem 3. Let $s(n) = n^2 + n$ be G_{nb}'s input length, and let $\mathsf{D_f}$ and $g = g^f$ be as in Definition 10. We prove that G_{nb}'s next-block pseudoentropy is at least $s(n) + \log n - 2$, where the proof that it is larger than $s(n) + c \cdot \log n$ for any $c > 0$ follows along similar lines. Let $\ell = \ell(n) = 2\log n$ and assume for simplicity that $\log n \in \mathbb{N}$. The one-wayness of f guarantees that $\mathsf{D_f}(f(x)) \leq n - \ell$ for all sufficiently large n and every $x \in \{0,1\}^n$; otherwise, the trivial inverter that returns a uniform element in $\{0,1\}^n$ inverts f with nonnegligible probability.

Define g' over $\{0,1\}^n \times \mathcal{H}_{n,n-\ell}$, by $g'(x,h) = (f(x), h, h(x)_{1,\ldots,\mathsf{D_f}(f(x))})$ (i.e., we have removed the last ℓ rows from the matrix defining the hash function h). The above observation about f yields that g' is well defined, and the hardness to invert of g (Claim 4) yields by a simple reduction that g' is also hard to invert.

Since g' is hard to invert, Lemma 9 yields that

$$\begin{aligned}(f(U_n), H_{n,n-\ell}, H_{n,n-\ell}(U_n)_{1,\ldots,\mathsf{D_f}(f(U_n))}, H'_{n,\ell}, H'_{n,\ell}(U_n)) &\equiv (g'(U_n, H_{n,n-\ell}), H'_{n,\ell}, H'_{n,\ell}(U'_\ell)) \\ &\approx_{\mathsf{nu\text{-}C}} (g'(U_n, H_{n,n-\ell}), H'_{n,\ell}, U'_\ell),\end{aligned}$$

where U_n and U'_ℓ are uniformly and independently distributed over $\{0,1\}^n$ and $\{0,1\}^\ell$, respectively, and $H_{n-\ell}$ and $H'n,\ell$ are uniformly and independently distributed over $\mathcal{H}_{n,n-\ell}$ and $\mathcal{H}_{n,\ell}$, respectively. Changing the order in the above and noting that $H_n \equiv (H_{n,n-\ell}, H_{n,\ell})$, yields that

$$(f(U_n), H_n, H_n(U_n)_{1,\ldots,\mathsf{D_f}(f(U_n))+\ell}) \approx_{\mathsf{nu\text{-}C}} (f(U_n), H_n, H_n(U_n)_{1,\ldots,\mathsf{D_f}(f(U_n))}, U'_\ell). \tag{.3}$$

Let $t(n) = 2n + n^2 = s(n) + n = G_{\mathrm{nb}}$'s output length. Let $X = X(n) = G_{\mathrm{nb}}(U_n, H_n)$, let $J = J(n) = s(n) + \mathsf{D_f}(f(U_n))$, and let $Y = Y(n) = (Y_1, \ldots, Y_m)$ be defined by $Y_i = X_i$ if $i \notin [J+1, J+\ell]$, and Y_i is set to a uniform bit otherwise (i.e., $i \in [J+1, J+\ell]$). Equation (.3) yields that $X_{J+1,\ldots,J+\ell}$ is computationally indistinguishable from U_ℓ given $X_{1,\ldots,J}$ and J, yielding that

$$(J, X_{\leq J+r}) \approx_{\mathsf{nu\text{-}C}} (J, X_{<J+r}, U) \tag{.4}$$

for every $r \in [\ell]$, where U is a uniform bit. It follows that, for every $\mathsf{ppt}^{\mathsf{NU}}$ D and $i \in [m]$, it holds that

[10] [7] states that $H_{n,\ell(n)}(U_n)$ is *computationally unpredictable* from $(v(U_n), H_{n,\ell(n)})$, but since $\left|H_{n,\ell(n)}(U_n)\right| \in O(\log n)$, the reduction to the above statement is standard.

$$\begin{aligned}
&\Pr\left[\mathsf{D}(1^n, X_{\le i}) = 1\right] - \Pr\left[\mathsf{D}(1^n, X_{<i}, Y_i) = 1\right] \qquad (.5)\\
&= \Pr\left[\mathsf{D}(1^n, X_{\le i}) = 1 \wedge i \notin [J, J+\ell]\right] - \Pr\left[\mathsf{D}(1^n, X_{<i}, Y_i) = 1 \wedge i \notin [J, J+\ell]\right]\\
&\quad + \sum_{r=1}^{\ell} \left(\Pr\left[\mathsf{D}(1^n, X_{\le i}) = 1 \wedge i = J+r\right] - \Pr\left[\mathsf{D}(1^n, X_{<i}, Y_i) = 1 \wedge i = J+r\right]\right)\\
&= 0 + \sum_{r=1}^{\ell} \left(\Pr\left[\mathsf{D}(1^n, X_{\le i}) = 1 \wedge i = J+r\right] - \Pr\left[\mathsf{D}(1^n, X_{<i}, Y_i) = 1 \wedge i = J+r\right]\right)\\
&\le 0 + \ell \cdot \mathrm{neg}(n)\\
&= \mathrm{neg}(n).
\end{aligned}$$

The second equality holds since $Y_i = X_i$ for $i \notin [J, J+\ell]$. The inequality holds since, if $\Pr\left[\mathsf{D}(1^n, X_{\le i}) = 1 \wedge i = J+r\right] - \Pr\left[\mathsf{D}(1^n, X_{<i}, Y_i) = 1 \wedge i = J+r\right] > \mathrm{neg}(n)$ for some i and r, then the nonuniform distinguisher D' that on input (j, x) returns $\mathsf{D}(x)$ if $j = i + r$, and a uniform bit otherwise, contradicts Equation (.4).

It is left to prove that Y has high entropy given the blocks of X. We compute

$$\begin{aligned}
\sum_{i=1}^{m} \mathrm{H}(Y_i \mid X_{<i}) &\ge \sum_{i=1}^{m} \mathrm{H}(Y_i \mid X_{<i}, J)\\
&= \mathop{\mathrm{E}}_{j \overset{\mathrm{R}}{\leftarrow} J}\left[\sum_{i=1}^{m} \mathrm{H}(Y_i \mid X_{<i}, J = j)\right]\\
&\ge \mathop{\mathrm{E}}_{j \overset{\mathrm{R}}{\leftarrow} J}\left[\sum_{i=1}^{j} \mathrm{H}(Y_i \mid X_{<i}, J = j) + \sum_{i=j+1}^{j+\ell} \mathrm{H}(Y_i \mid X_{<i}, J = j)\right]\\
&= \mathop{\mathrm{E}}_{j \overset{\mathrm{R}}{\leftarrow} J}\left[\sum_{i=1}^{j} \mathrm{H}(X_i \mid X_{<i}, J = j) + \sum_{i=j+1}^{j+\ell} 1\right]\\
&= \mathop{\mathrm{E}}_{j \overset{\mathrm{R}}{\leftarrow} J}\left[\mathrm{H}(X_{\le j} | J = j)\right] + \ell\\
&= \mathrm{H}(X_{\le J} | J) + \ell.
\end{aligned}$$

It follows that

$$\begin{aligned}
\sum_{i=1}^{m} \mathrm{H}(Y_i \mid X_{<i}) &\ge \ell + \mathrm{H}(X_{\le J}) - \mathrm{H}(J)\\
&\ge \ell + \mathrm{H}(X_{\le J}) - \log n\\
&\ge \ell + s(n) - 2 - \log n\\
&= s(n) + \log n - 2.
\end{aligned}$$

The penultimate inequality follows by Claim 4 (since $\mathrm{H}(X_{\le J}) \ge \mathrm{H}_2(X_{\le J}) = \log(1/\mathrm{CP}(X_{\le J})) \ge s(n) - 2$). We conclude that Y realizes the claimed next-block pseudoentropy of G_{nb}. □

Proving Claim 4.

Proof of Claim 4. Let (U_n', H_n') be an independent copy of (U_n, H_n). Then

$$\begin{aligned}
&\mathrm{CP}(g(U_n, H_n)) \\
&= \Pr\left[g(U'_n, H'_n) = g(U_n, H_n)\right] \\
&= \Pr\left[(f(U'_n), H'_n, H'_n(U'_n)_{1,\ldots,\mathsf{D_f}(f(U'_n))}) = (f(U_n), H_n, H_n(U_n)_{1,\ldots,\mathsf{D_f}(f(U_n))})\right] \\
&= \mathop{\mathrm{E}}_{y \overset{\mathrm{R}}{\leftarrow} f(U_n)}\left[\Pr\left[f(U'_n) = y\right] \cdot \Pr\left[H'_n = H_n\right] \cdot \Pr\left[H_n(U'_n)_{1,\ldots,\mathsf{D_f}(y)} = H_n(U_n)_{1,\ldots,\mathsf{D_f}(y)} \mid f(U'_n) = y\right]\right] \\
&\le \mathop{\mathrm{E}}_{y \overset{\mathrm{R}}{\leftarrow} f(U_n)}\left[\frac{2^{\mathsf{D_f}(y)}}{2^n} \cdot \frac{1}{|\mathcal{H}_n|} \cdot \left(\frac{1}{2^{\mathsf{D_f}(y)-1}} + \frac{1}{2^{\mathsf{D_f}(y)}}\right)\right] \\
&\le \frac{3}{|\{0,1\}^n \times \mathcal{H}_n|}.
\end{aligned}$$

The first inequality holds since $\Pr\left[H_n(U'_n)_{1,\ldots,\mathsf{D_f}(y)} = H_n(U_n)_{1,\ldots,\mathsf{D_f}(y)} \mid f(U'_n) = y\right]$ is upper bounded by $\Pr\left[U'_n = U'_n \mid f(U'_n) = y\right] + \Pr\left[H_n(x)_{1,\ldots,\mathsf{D_f}(y)} = H_n(x')_{1,\ldots,\mathsf{D_f}(y)}\right]$ for some $x \neq x'$. □

Proving Claim 5.

Proof of Claim 5. This fact was first proven in [17] using the leftover hash lemma (Lemma 8). Here, we present a different proof that is inspired by Rackoff's proof of the Leftover Hash Lemma, and uses the high collision probability of g directly.

Let $\mathrm{Inv_g}$ be a nonuniform polynomial-time algorithm that inverts $g(U_n, H_n)$ with probability $\delta = \delta(n)$. We show that there exists an inverter Inv that inverts f with probability at least roughly δ^2/n, from which the claim follows.

Fix $n \in \mathbb{N}$, and let $\mathcal{L} \subseteq \mathrm{Im}(g(\{0,1\}^n \times \mathcal{H}_n))$ be the set of outputs where $\mathrm{Inv_g}$ inverts g correctly (without loss of generality $\mathrm{Inv_g}$ is deterministic). By assumption, $\Pr\left[g(U_n, H_n) \in \mathcal{L}\right] = \delta$. Since the collision probability of a distribution is at least the reciprocal of its support size, it follows that

$$\begin{aligned}
\mathrm{CP}(g(U_n, H_n)) &= \Pr\left[g(U_n, H_n) = g(U'_n, H'_n)\right] \\
&\ge \Pr\left[g(U_n, H_n), g(U'_n, H'_n) \in \mathcal{L}\right] / |\mathcal{L}| \\
&= \delta^2 / |\mathcal{L}|.
\end{aligned}$$

By Claim 4, $\mathrm{CP}(g(U_n, H_n)) \le 3 \cdot \frac{1}{|\mathcal{H}_n|} \cdot \frac{1}{2^n}$, and therefore

$$\frac{3 \cdot |\mathcal{L}|}{|\mathcal{H}_n| \cdot 2^n} \ge \delta^2. \qquad (.6)$$

Now for $y \in \mathrm{Im}(f(\{0,1\}^n))$, let $\mathcal{L}_y = \{(h, z) \colon (y, h, z) \in \mathcal{L}\}$. It follows that

$$\begin{aligned}
\Pr\left[(f(U_n), H_n, U'_{\mathsf{D_f}(f(U_n))}) \in \mathcal{L}\right] &= \mathop{\mathrm{E}}_{y \overset{\mathrm{R}}{\leftarrow} f(U_n)}\left[\Pr\left[(H_n, U'_{\mathsf{D_f}(y)}) \in \mathcal{L}_y\right]\right] \qquad (.7)\\
&= \mathop{\mathrm{E}}_{y \overset{\mathrm{R}}{\leftarrow} f(U_n)}\left[\frac{|\mathcal{L}_y|}{|\mathcal{H}_n| \times 2^{\mathsf{D_f}(y)}}\right]\\
&= \sum_{y \in \mathrm{Im}(f)} \frac{\left|f^{-1}(y)\right|}{2^n} \cdot \frac{|\mathcal{L}_y|}{|\mathcal{H}_n| \times 2^{\mathsf{D_f}(y)}}\\
&\geq \sum_{y \in \mathrm{Im}(f)} \frac{2^{\mathsf{D_f}(y)-1}}{2^n} \cdot \frac{|\mathcal{L}_y|}{|\mathcal{H}_n| \times 2^{\mathsf{D_f}(y)}}\\
&= \frac{1}{|\mathcal{H}_n| \cdot 2^{n+1}} \cdot \sum_y |\mathcal{L}_y|\\
&= \frac{1}{|\mathcal{H}_n| \cdot 2^{n+1}} \cdot |\mathcal{L}|\\
&\geq \delta^2/6.
\end{aligned}$$

Consider the following (randomized) inverter for f:

Algorithm 6 (Inv)**.**

Oracle: $\mathrm{Inv_g}$
Input: $y \in \{0,1\}^n$

```
1. Let $h \overset{R}{\leftarrow} \mathcal{H}_n$, $i \overset{R}{\leftarrow} [n]$, and $z \overset{R}{\leftarrow} \{0,1\}^i$.
2. Let $(x, h') = \mathrm{Inv_g}(y, h, z)$.
3. Return $x$.
```

Let I be the random variable corresponding to the value of $i \overset{\mathrm{R}}{\leftarrow} [y]$ in the execution of $\mathrm{Inv}(y)$. Compute

$$\begin{aligned}
&\Pr\left[\mathrm{Inv}(f(U_n)) \in f^{-1}(f(U_n))\right]\\
&\geq \Pr\left[I = \mathsf{D_f}(f(U_n))\right] \cdot \Pr\left[\mathrm{Inv}(f(U_n)) \in f^{-1}(f(U_n)) \mid I = \mathsf{D_f}(f(U_n))\right]\\
&= \frac{1}{n} \cdot \Pr\left[\mathrm{Inv_g}(f(U_n), H_n, U_{\mathsf{D_f}(f(U_n))}) \in g^{-1}(f(U_n), H_n, U_{\mathsf{D_f}(f(U_n))})\right]\\
&\geq \frac{1}{n} \cdot \delta^2/4 = \delta^2/6n.
\end{aligned}$$

The first inequality holds since I is independent of y, and the second inequality is by Equation (.7). It follows that there exists a nonuniform polynomial-time algorithm that inverts f with probability at least $\delta(n)^2/6n$, implying that that $\delta(n) = \mathrm{neg}(n)$. □

3.3 Manipulating Next-Block Pseudoentropy

In this section we develop tools to manipulate next-block pseudoentropy. These tools are later used in Section 3.4 to convert the next-block pseudoentropy constructed in Section 3.2 into a pseudorandom generator.

The tools considered below are rather standard "entropy manipulations": entropy equalization (i.e., picking a random variable at random from a set of random variables to get a new random variable whose entropy is the average entropy), parallel repetition, and extraction from high-min-entropy sources, and their effect on the real entropy of random variables is clear. Fortunately, these manipulations have essentially the same effect also on the next-block pseudoentropy of a random variable. In Section 4.2, we show that these manipulations also have the desired effect on the *accessible entropy* of a random variable, a similarity that implies the similarity between the pseudorandom generator construction presented in this section, and the construction of statistically hiding commitment scheme, presented in Section 4.2.

3.3.1 Entropy Equalization via Truncated Sequential Repetition

This manipulation takes independent copies of an m-block random variable with next-block pseudoentropy at least k and concatenates them. It then truncates, at random, some of the first and final output blocks of the concatenated variable. The effect of this manipulation is that *each block* of the resulting variable has next-block pseudoentropy at least k/m. This per-block knowledge of the next-block pseudoentropy becomes very handy for constructing pseudorandom generators.

The price of this manipulation is that we "give away" some next-block pseudoentropy, but when taking enough copies, this loss is not significant.

Definition 12. *For* $\mathbf{z} = (z_1, \ldots, z_t)$ *and* $1 \leq j \leq m \leq t$, *let* $\mathrm{Equalizer}^m(j, \mathbf{z}) := z_j, \ldots, z_{t+j-m-1}$.

That is, $\mathrm{Equalizer}^m(j, \mathbf{z})$ removes the first $(j-1)$ and last $(m-j+1)$ elements from $\mathbf{z}$.

Lemma 10. *Let* $m = m(n)$ *be a power of* 2,[11] *assume* $X = X(n) = (X_1, \ldots, X_m)$ *has next-block pseudoentropy (at least)* $k = k(n)$, *and let* $w = w(n) \geq 2$ *be a polynomially bounded integer function. Let* $X^{[w]} = X^{[w]}(n)$ *be the* $(m \cdot (w-1))$*-block random variable defined by* $X^{[w]}(n) = \mathrm{Equalizer}^m(J, X^{(w)})$, *where* $J = J(n)$ *is uniformly distributed over* $[m(n)]$, *and* $X^{(w)} = X^{(w)}(n) = (X^1, \ldots, X^w)$, *for* $X^1, \ldots, X^w$ *being independent copies of* $X(n)$. *Then every block of* $X^{[w]}$ *has next-block pseudoentropy (at least)* k/m.

Namely, the next-block pseudoentropy of each block of $X^{[w]}$ is the *average* next-block pseudoentropy of the blocks of X.

Proof. Let $Y = Y(n) = (Y_1, \ldots, Y_m)$ be a random variable that realizes the next-block pseudoentropy of X, and let $Y^{(w)} = Y^{(w)}(n) = (Y^1, \ldots, Y^w)$ be jointly distributed with $X^{(w)} = (X^1, \ldots, X^w)$ in the natural way—Y^j is jointly distributed with X^j according to the joint distribution (X, Y). We prove that $Y^{[w]} = Y^{[w]}(n) = \mathrm{Equalizer}^m(J, Y^{(w)})$ realizes the claimed per-block next-block pseudoentropy of $X^{[w]}$. In the following we let $\tilde{m} = \tilde{m}(n) = (w-1) \cdot m$.

We start by proving that each block of $Y^{[w]}$ has high entropy given the previous blocks of $Y^{[w]}$. Fix $n \in \mathbb{N}$ and omit that from the notation, and fix $i \in [\tilde{m}]$. By chain rule for Shannon entropy, it holds that

$$\begin{aligned}\mathrm{H}(Y_i^{[w]} \mid X_{<i}^{[w]}) &\geq \mathrm{H}(Y_{i+J-1}^{(w)} \mid X_{<i+J-1}^{(w)}, J) \\ &= \mathrm{H}(Y_{i+J-1 \bmod m} \mid X_{<i+J-1 \bmod m}),\end{aligned} \tag{.8}$$

letting $m \bmod m$ be m (rather than 0). The equality follows from the fact that, for any $t \in [mw]$, $(Y_t^{(w)}, X_{t-1}^{(w)}, \ldots, X_{t'=\lfloor t/m \rfloor \cdot m+1}^{(w)})$ is independent of $X_{<t'}^{(w)}$, and is identically distributed to $(Y_{t \bmod m}, X_{<t \bmod m})$.

Since $(i + J - 1 \bmod m)$ is uniformly distributed in $[m]$, it follows that

$$\begin{aligned}\mathrm{H}(Y_{i+J-1 \bmod m} \mid X_{<i+J-1 \bmod m}, J) &= \mathrm{E}_{i' \xleftarrow{R} [m]}[\mathrm{H}(Y_{i'} \mid X_{<i'})] \\ &= \frac{1}{m} \cdot \sum_{i' \xleftarrow{R} [m]} \mathrm{H}(Y_{i'} \mid X_{<i'}) \\ &\geq k/m,\end{aligned} \tag{.9}$$

and we conclude that $\mathrm{H}(Y_i^{[w]} \mid X_{<i}^{[w]}) \geq k/m$ for every $i \in [\tilde{m}]$.

For the second part, let D be a $\mathsf{ppt}^{\mathsf{NU}}$, let $i = i(n) \in [\tilde{m}(n)]$, and let

[11] Any other restriction that allows an efficient sampling from $[m(n)]$ will do.

$$\varepsilon_{\mathsf{D}}(n) := \Pr\left[\mathsf{D}(1^n, X^{[w]}(n)_{\leq i}) = 1\right] - \Pr\left[\mathsf{D}(1^n, X^{[w]}(n)_{<i}, Y^{[w]}(n)_i) = 1\right] \tag{.10}$$

In the following we omit n whenever clear from the context. A similar argument to that used in the first part yields that

$$\begin{aligned}
\varepsilon_{\mathsf{D}}(n) &= \Pr\left[\mathsf{D}(X^{[w]}_{\leq i}) = 1\right] - \Pr\left[\mathsf{D}(X^{[w]}_{<i}, Y^{[w]}_i) = 1\right] \qquad (.11)\\
&= \Pr\left[\mathsf{D}(X^{w}_{J,\ldots,i+J-1}) = 1\right] - \Pr\left[\mathsf{D}(X^{w}_{J,\ldots,i+J-2}, Y^{w}_{\ldots,i+J-1}) = 1\right]\\
&= \Pr\left[\mathsf{D}(X^{w-1}_{J,\ldots,(w-1)m}, X_{\leq i+J-1 \bmod m}) = 1\right] - \Pr\left[\mathsf{D}(X^{w-1}_{J,\ldots,(w-1)m}, X_{<i+J-1 \bmod m}, Y_{i+J-1 \bmod m}) = 1\right]\\
&\leq \Pr\left[\mathsf{D}(x, X_{\leq i+j-1 \bmod m}) = 1\right] - \Pr\left[\mathsf{D}(x, X_{<i+j-1 \bmod m}, Y_{i+j' \bmod m}) = 1\right]
\end{aligned}$$

for some fixing of $j \in [m]$ and $x \in \operatorname{Supp}(X^{w-1}_{j,\ldots,(w-1)m})$. Hence, there exists a $\mathsf{ppt}^{\mathsf{NU}}$ D' such that

$$\varepsilon_{\mathsf{D}'}(n) := \Pr\left[\mathsf{D}'(X_{\leq i'}) = 1\right] - \Pr\left[\mathsf{D}'(X_{<i'}, Y_{i'}) = 1\right] \geq \varepsilon_{\mathsf{D}}(n) \tag{.12}$$

for some $i' = i'(n) \in [m(n)]$. Since Y is block-wise indistinguishable from X, it follows that $\varepsilon_{\mathsf{D}'}(n) = \operatorname{neg}(n)$ and therefore $\varepsilon_{\mathsf{D}}(n) = \operatorname{neg}(n)$. Hence, $Y^{[w]}$ is block-wise indistinguishable from $X^{[w]}$. □

3.3.2 Parallel Repetition

This manipulation, which simply takes parallel repetition (i.e., direct product) of a random variable, has a twofold effect. The first is that the overall next-block pseudoentropy a t-fold parallel repetition of a random variable X is t times the next-block pseudoentropy of X. Hence, if X's next-block pseudoentropy is larger than the number of bits it takes to sample it, this gap gets multiplied by t in the resulting random variable. The second effect of taking such a product is turn next-block pseudoentropy into next-block pseudo-*min-entropy.*

Lemma 11. *Let $m = m(n)$ and $\ell = \ell(n)$ be integer functions, assume every block of $X = X(n) = (X_1, \ldots, X_m)$ is of length $\ell(n)$ and has next-block pseudoentropy (at least) $\alpha = \alpha(n)$, and let $t = t(n)$ be polynomially bounded integer function. Let $X^{\langle t\rangle} = X^{\langle t\rangle}(n)$ be the m-block random variable defined by $X^{\langle t\rangle} = X^{\langle t\rangle}(n) = \big((X_1^1, \ldots, X_1^t), \ldots, (X_m^1, \ldots, X_m^t)\big)$, for $X^1, \ldots, X^t$ being independent copies of X. Then every block of $X^{\langle t\rangle}$ has next-block pseudo-min-entropy (at least) $\alpha'(n) = t \cdot \alpha - O(\log n \cdot (\ell + \log n) \cdot \sqrt{t})$.*

Notice that the $t \cdot \alpha$ term in the above statements 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. For instance, consider the case that X has next-block pseudoentropy at least α and $\ell = 1$ (i.e., X is a sequence of bits), and that we would like to deduce that $X^{\langle t\rangle}$ has next-block pseudo-min-entropy $\alpha' = t \cdot (\alpha - \delta)$ for some $\delta > 0$. Lemma 11 guarantees that this happens for $t = \operatorname{polylog}(n)/\delta^2$.

Proof. Let $Y = Y(n) = (Y_1, \ldots, Y_m)$ be a random variable that realizes the per-block next-block pseudoentropy of X, and let $Y^{\langle t\rangle} = Y^{\langle t\rangle}(n) = \big((Y_1^1, \ldots, Y_1^t), \ldots, (Y_m^1, \ldots, Y_m^t)\big)$ be jointly distributed with $X^{\langle t\rangle}$ in the natural way—Y^j is jointly distributed with X^j according to the joint distribution (X, Y). Since Y is block-wise indistinguishable from X, and since $t(n) \leq \operatorname{poly}(n)$, a straightforward hybrid argument yields that $Y^{\langle t\rangle}$ is block-wise indistinguishable from $X^{\langle t\rangle}$.

Since $\mathrm{H}(Y_i \mid X_{<i}) \geq \alpha$ for every $i \in [m]$, applying Lemmas 3 and 5 with $\varepsilon = 2^{-\log^2 n}$ yields that there exists a random variable $W = W(n) = (W_1, \ldots, W_m)$ jointly distributed with $X^{\langle t\rangle}$, such that the following hold for every $i = i(n) \in [m(n)]$:

1. $\mathit{\Delta}((X_{<i}^{\langle t\rangle}, Y_i^{\langle t\rangle}), (X_{<i}^{\langle t\rangle}, W_i)) = \operatorname{neg}(n)$, and
2. $\mathrm{H}_\infty(W_i|_{X_{<i}^{\langle t\rangle}=x_{<i}}) \geq \alpha - O((\log n + \ell)\cdot \log n \cdot \sqrt{t})$, for every $x \in \operatorname{Supp}(X_{<i}^{\langle t\rangle})$.

Item 1 and the previous observation yield that W is block-wise indistinguishable from $X^{\langle t\rangle}$, and by item 2 we conclude that W realizes the claimed next-block pseudo-min-entropy of $X^{\langle t\rangle}$. $\square$

3.3.3 Block-wise Extraction

The tool applies a randomness extractor separately to each of the random variable blocks, to convert per-block next-block pseudo-min-entropy into pseudorandomness. The result is a sufficiently long pseudorandom sequence. This is a computational analogue of block-source extraction in literature on randomness extractors [5, 33]. The price of this manipulation is that the length, and thus the amount of pseudorandomness, of the resulting variable is shorter than the overall pseudoentropy of the original variable, due to inherent entropy loss in randomness extraction.

Lemma 12. *Let $m = m(n)$ and $\ell = \ell(n)$ be integer functions, and assume that every block of $X = X(n) = (X_1, \ldots, X_m)$ over $(\{0,1\}^\ell)^m$ has next-block pseudoentropy (at least) $\alpha = \alpha(n) \geq \lceil \log^2 n \rceil$. Then there exists a polynomial-time computable* $\operatorname{Ext}\colon \{0,1\}^\ell \times (\{0,1\}^\ell)^m \mapsto \left(\{0,1\}^{\lfloor\alpha\rfloor - \lceil \log^2 n\rceil}\right)^m$ *such that* $(R, \operatorname{Ext}(R,X))$, *for $R = R(n) \overset{R}{\leftarrow} \{0,1\}^\ell$, is pseudorandom.*

Proof. Let $\beta = \beta(n) = \lfloor\alpha\rfloor - \lceil \log^2 n \rceil$. For $r, x \in \{0,1\}^\ell$, let $h_r(x) := r\cdot x$ over $\mathrm{GF}(2^\ell)$, truncated to the first β bits. Note that $\{h_r\colon r \in \{0,1\}^\ell\}$ is a two-universal hash family over $\{0,1\}^\ell$. For $x = (x_1, \ldots, x_k) \in (\{0,1\}^\ell)^k$, let $\operatorname{Ext}(r,x) = (h_r(x_1), \ldots, h_r(x_k))$. Let $\mathsf{D}^{\mathrm{PRG}}$ be $\mathsf{ppt}^{\mathsf{NU}}$, and assume that

$$\varepsilon(n) := \Pr\left[\mathsf{D}_{\mathrm{PRG}}(1^n, R, \operatorname{Ext}(X,R)) = 1\right] - \Pr\left[\mathsf{D}_{\mathrm{PRG}}(1^n, R, U_{m\cdot\beta}) = 1\right] \neq \operatorname{neg}(n). \tag{.13}$$

In the following we omit n whenever clear from the context. A hybrid argument yields that there exists $i \in [m]$ and $\mathsf{ppt}^{\mathsf{NU}}$ D such that

$$\Pr\left[\mathsf{D}(R, \operatorname{Ext}(R, X_{\leq i})) = 1\right] - \Pr\left[\mathsf{D}(R, \operatorname{Ext}(R, X_{<i}), U_\beta) = 1\right] \geq \varepsilon/m.$$

Let $Y = Y(n)$ be a random variable that realizes the per-block next-block **pseudo-min-entropy** of X. Since $\mathrm{H}_\infty(Y_i|_{X_{<i}} = x_{<i}) \geq \alpha$ for every $x \in \operatorname{Supp}(X)$, and since $\{h_r\colon r \in \{0,1\}^\ell\}$ is a two-universal hash family, the leftover hash lemma (Lemma 8) yields that

$$\operatorname{SD}((R, h_R(Y_i|_{X_{<i}} = x_{<i})), (R, U_\beta)) \leq 2^{-\log^2 n}$$

for every $x \in \operatorname{Supp}(X)$. It follows that

$$\operatorname{SD}(R, \operatorname{Ext}(R, X_{<i}, Y_i), (R, \operatorname{Ext}(R, X_{<i}), U_\beta)) \leq 2^{-\log^2 n}$$

and therefore

$$\begin{aligned} &\Pr\left[\mathsf{D}(R, \operatorname{Ext}(R, X_{\leq i})) = 1\right] - \Pr\left[\mathsf{D}(R, \operatorname{Ext}(R, X_{<i}, Y_i)) = 1\right] \\ &\quad \geq \varepsilon/\ell - 2^{-\log^2 n} \neq \operatorname{neg} \end{aligned} \tag{.14}$$

For $n \in \mathbb{N}$, let $r_n \in \operatorname{Supp}(R)$ be the string that maximizes the above gap, and consider the distinguisher D' that on input $(1^n, z)$, returns $\mathsf{D}(1^n, r_n, \operatorname{Ext}(r_n, z))$. Equation (.14) yields that

$$\Pr\left[\mathsf{D}'(X_{\leq i}) = 1\right] - \Pr\left[\mathsf{D}'(X_{<i}, Y_i) = 1\right] \neq \operatorname{neg}$$

Hence, the $\mathsf{ppt}^{\mathsf{NU}}$ D' contradicts the assumed block-wise indistinguishability of Y from X. $\square$

3.4 Putting It Together: One-Way Functions to Pseudorandom Generators

In this section we use the results of previous sections to construct pseudorandom generators from next-block pseudoentropy generators.

It is clear that a pseudorandom generator from n bits to $m(n) > n$ has next-block pseudoentropy $m(n)$, hence, it is a next-block entropy generator with entropy gap $(m(n) - n)$—its next-block pseudoentropy is larger than its real entropy by $(m(n) - n)$. The following theorem provides the converse direction.

Theorem 7 (Next-block pseudoentropy to pseudorandom generator). *For any polynomial-time computable and polynomially bounded integer function $s = s(n)$ and polynomial-time computable function $\Delta = \Delta(n) \le 2$, there exists a polynomial-time computable integer function $s' = s'(n) = \Theta(s \cdot \Delta^{-3} \cdot \mathrm{polylog}(n))$ such that the following holds: Assuming there exists a polynomial-time generator $G_{\mathrm{nb}} \colon \{0,1\}^s \mapsto \{0,1\}^{2s}$ with next-block pseudoentropy $s(1+\Delta)$, then there exists a pseudorandom generator $G \colon \{0,1\}^{s'} \mapsto \{0,1\}^{s' \cdot (1+\Theta(\Delta))}$. Furthermore, G uses G_{nb} as an oracle (i.e., black box) and on inputs of length s', all calls of G to G_{nb} are on inputs of length s.*

Proof. The proof is done by manipulating the next-block pseudoentropy of G_{nb} using the tools described in Section 3.3.

Let $X = X(n) = G_{\mathrm{nb}}(U_s)$. We assume without loss of generality that, for every n, the number $m(n) = 2s(n)$ of output blocks (=bits) of G_{nb} is a power of 2 (by padding with zeros if necessary). By assumption, X has next-block pseudoentropy $s(1 + \Delta)$.

Truncated sequential repetition: pseudoentropy equalization. The first step is to use X to define a random variable $X^{[w]}$ that *each of whose blocks* has the same amount of next-block pseudoentropy— the average of the next-block pseudoentropy of the blocks of X. The entropy gap of $X^{[w]}$, in relative terms, is essentially the same as that of X.

For $w = w(n) = \lceil 8/\Delta \rceil$, let $X^{[w]} = X^{[w]}(n)$ be the truncated sequential repetition of X according to Lemma 10. Namely, $X^{[w]}$ consists of w independent copies of X, omitting the first $(J-1)$ blocks of the first copy and the last $(m-J+1)$ blocks of the last copy, for $J \overset{\mathrm{R}}{\leftarrow} [m]$. Note that $X^{[w]}$ can be generated efficiently using $s' = s'(n) = \log(m) + w \cdot s$ random bits, and has $m' = m'(n) = m(w-1)$ blocks.

By Lemma 10, each block of $X^{[w]}$ has next-block pseudoentropy $\alpha = \alpha(n) = s(1+\Delta)/m = \frac{1}{2} + \Delta/2$.

Parallel repetition: converting Shannon pseudoentropy to pseudo-min-entropy and gap amplification. In this step $X^{[w]}$ is used to construct a random variable $(X^{[w]})^{\langle t \rangle}$ that each of whose blocks has the same amount of *pseudo-min-entropy*—about t time the per-block pseudoentropy of $X^{[w]}$.

For $t = t(n) = \lceil \log^5 n \cdot \Delta^{-2} \rceil$, let $(X^{[w]})^{\langle t \rangle} = (X^{[w]})^{\langle t \rangle}(n)$ be the t-fold parallel repetition of $X^{[w]}$ (see Lemma 11). That is, the i-th block of $(X^{[w]})^{\langle t \rangle}$, contains the i-th blocks of the t independent copies of $X^{[w]}$. Note that $(X^{[w]})^{\langle t \rangle}$ can be generated efficiently using $s'' = s''(n) = t \cdot s'$ bits, and has m' blocks.

By Lemma 11, each block of $(X^{[w]})^{\langle t \rangle}$ has next-block pseudo-min-entropy $\alpha' = \alpha'(n) = t \cdot \alpha - O(\log^2 n \cdot \sqrt{t})$, which is larger than $t \cdot (\frac{1}{2} + \Delta/4)$ for large enough n.

Randomness extraction: converting pseudo-min-entropy to pseudorandomness. In the final step, pseudorandom bits are extracted from $(X^{[w]})^{\langle t \rangle}$, by applying a randomness extractor on each of its blocks.

Lemma 12 yields that there exists an efficient $\mathrm{Ext} \colon \{0,1\}^{t(n)} \times (\{0,1\}^t)^{m'} \mapsto \left(\{0,1\}^{\alpha' - \lceil \log^2 n \rceil}\right)^{m'}$ such that $X^{\mathrm{PRG}} = X^{\mathrm{PRG}}(n) = (R, \mathrm{Ext}(R, (X^{[w]})^{\langle t \rangle})$, for $R = R(n) \overset{\mathrm{R}}{\leftarrow} \{0,1\}^t$, is pseudorandom. We remind the reader that $\mathrm{Ext}(r, x = (x_1, \ldots, x_m))$ merely applies (the same) two-universal function h_r on each of x's blocks. Note that it takes s''' bits to efficiently sample X^{PRG}, for

$$s''' = s'''(n) = t + s'' = t(\ell s + \Theta(\log n)) = \Theta(s\Delta^{-3} \cdot \mathrm{polylog}(n)).$$

It follows that, for large enough n, the length of $X^{\mathrm{PRG}}(n)$ is at least

$$\begin{aligned}
m'(\alpha' - \lceil \log^2 n \rceil) &\geq m'(t\alpha - O(\log^3 n \cdot \sqrt{t})) \\
&> m'(t(\frac{1}{2} + \Delta/2) - O(\log^3 n \cdot \sqrt{t})) \\
&> m't(\frac{1}{2} + \Delta/4) \\
&= s(\ell - 1)t(1 + \Delta/2) \\
&\geq s\ell t(1 + \Delta/4) \\
&\geq s'''(1 + \Delta/8).
\end{aligned}$$

Hence, $\left|X^{\mathrm{PRG}}\right| = s'''(1 + \Omega(\Delta))$, and since all the above manipulations were efficient, the proof of the theorem follows. □

Remark 3 (Tighter reduction). Vadhan and Zheng [29] noticed that, by modifying the construction used in the proof of Theorem 7, one can construct an efficient generator G and a random variable $Z = Z(n)$ such that the following hold:

1. It takes $s'(n) = \Theta(s(n) \cdot \Delta(n)^{-2} \cdot \operatorname{polylog}(n))$ bits to efficiently sample Z, i.e., a factor of Δ^{-1} shorter than the input length of the pseudorandom generator in Theorem 7.
2. $G(Z)$ is computationally indistinguishable from (Z, U), where U is a random string of length $\Omega(s(n)\Delta/n)$.

Then, by iterating G on its output (in a similar manner to the Blum-Micali pseudorandom generator length extending approach), without investing new randomness, they get a pseudorandom generator of seed length $s(n)$.

Combining the above Theorem 7 with Theorem 3 from the previous subsection yields the following result:

Theorem 8 (One-way function to pseudorandom generator). *There exists a polynomial-time computable function $s = s(n) = \Theta(n^7 \cdot \operatorname{polylog} n)$ such that the following holds: Let $f\colon \{0,1\}^n \mapsto \{0,1\}^n$ be nonuniformly one-way function, then there exists a pseudorandom generator $G\colon \{0,1\}^s \mapsto \{0,1\}^{s\cdot(1+\Omega(1/n^2))}$. Furthermore, G uses f as an oracle (i.e., black box) and on inputs of length $s(n)$, all calls of G to f are on inputs of length n.*

Proof. Pad the output of the next-block pseudoentropy generator guaranteed by Theorem 3 to make it length doubling (it is easy to see that this does not change its next-block pseudoentropy) and apply Theorem 7. □

Remark 4 (Tighter reduction, take 2). Plugging into Theorem 7 the next-block generators of Haitner et al. [14] or of Vadhan and Zheng [29], both with $s = \Theta(n)$ and $\Delta = \Theta(\log(n)/n)$, yields a pseudorandom generator of seed length $\Theta(n^4 \cdot \operatorname{polylog} n)$. If the latter generators are used with the tighter reduction of Vadhan and Zheng [29] mentioned above, the resulting generator has seed length $\Theta(n^3 \cdot \operatorname{polylog} n)$, which is the best we know how to achieve today.

4 Inaccessible Entropy and Statistically Hiding Commitment

In this section, we formally define the notion of inaccessible entropy and use it as intermediate tool to construct statistically hiding commitment from one-way functions.

We start, Section 4.1, by presenting the formal definition of inaccessible entropy. In Section 4.2, we show that any one-way function can be used to construct inaccessible entropy generator. In Section 4.3, we develop means to manipulate inaccessible entropy. Finally, in Section 4.4 we give a simplified version of the (still rather complicated) construction of statistically hiding commitments from inaccessible entropy generators.

4.1 Inaccessible Entropy Generators

We begin by informally recalling the definition from the introduction. Let $\mathsf{G}\colon \{0,1\}^n \mapsto (\{0,1\}^*)^m$ be an m-block generator over $\{0,1\}^n$ and let $\mathsf{G}(1^n) = (Y_1, \ldots, Y_m)$ denote the output of G over a uniformly random input. The *real entropy* of G is the (Shannon) entropy in G's output blocks, where for each block Y_i, we take its entropy conditioned on the previous blocks $Y_{<i} = (Y_1, \ldots, Y_{i-i})$. The *accessible entropy* of an arbitrary, adversarial m-block generator $\widetilde{G}$, with the same block structure as of G, is the entropy of the block of $\widetilde{G}$ conditioned not only on the previous blocks but also on the *coins used by $\widetilde{G}$ to generate the previous blocks.* The generator $\widetilde{G}$ is allowed to flip fresh random coins to generate its next block, and this is indeed the source of entropy in the block (everything else is fixed). We insist that the messages of $\widetilde{G}$ will be consistent with G: the support of $\widetilde{G}$'s messages is contained in that of G.

Moving to the formal definitions, we first define an m-block generator and then define the real and accessible entropy of such a generator.

Definition 13 (Block generators). *Let n be a security parameter, and let $m = m(n)$ and $s = s(n)$. An m-*block generator *is a function $\mathsf{G}\colon \{0,1\}^s \mapsto (\{0,1\}^*)^m$. The generator G is* efficient *if its running time on input of length $s(n)$ is polynomial in n.*

We call parameter n the security parameter*, s the* seed length*, m the* number of blocks*, and $\ell(n) = \max_{x\in\{0,1\}^{s(n)}, i\in[m(n)]} |\mathsf{G}(x)_i|$ the* maximal block length *of G.*

4.1.1 Real Entropy

Recall that we are interested in lower bounds on the real entropy of a block generator. We define two variants of real entropy: real Shannon entropy and real min-entropy. We connect these two notions through the notion of real sample-entropy. In other words, for a fixed m-tuple output of the generator, we ask how surprising were the blocks output by G in this tuple. We then get real Shannon entropy by taking the expectation of this quantity over a random execution and the min-entropy by taking the minimum (up to negligible statistical distance). An alternative approach would be to define the notions through the sum of conditional entropies (as we do in the intuitive description in the introduction). This approach would yield closely related definitions, and in fact exactly the same definition in the case of Shannon entropy (see Lemma 13).

Definition 14 (Real sample-entropy). *Let n be a security parameter, and let G be an m-block generator over $\{0,1\}^s$, for $m = m(n)$ and $s = s(n)$. For $i \in [m]$, define the* real sample-entropy of *$\mathbf{y} \in \operatorname{Supp}((Y_1, \ldots, Y_i) = \mathsf{G}(U_s)_{1,\ldots,i})$ as*

$$\mathrm{RealH}_{\mathsf{G}}(\mathbf{y}) = \sum_{j\in[i]} \mathrm{RealH}^j_{\mathsf{G}}(\mathbf{y})$$

for

$$\mathrm{RealH}^j_{\mathsf{G}}(\mathbf{y}) := \mathrm{H}_{Y_j|Y_{<j}}(\mathbf{y}_j|\mathbf{y}_{<j})$$

Definition 15 (Real entropy). *Let n be a security parameter, and let G be an m-block generator over $\{0,1\}^s$, for $m = m(n)$ and $s = s(n)$. We say that an m-block generator G has* real entropy at least $k = k(n)$*, if*

$$\mathop{\mathrm{E}}_{\mathbf{y}\xleftarrow{R}\mathsf{G}(U_s)} [\mathrm{RealH}_{\mathsf{G}}(\mathbf{y})] \geq k$$

for every $n \in \mathbb{N}$.

The generator G has real min-entropy at least k in its i-th block*, where $i = i(n) \in [m(n)]$, if*

$$\Pr_{\mathbf{y}\xleftarrow{R}\mathsf{G}(U_s)} \left[\mathrm{RealH}^i_{\mathsf{G}}(\mathbf{y}) < k\right] = \mathrm{neg}(n).$$

We observe that the real Shannon entropy simply amounts to measuring the standard conditional Shannon entropy of G's output blocks.

Lemma 13. *For an m-block generator G over $\{0,1\}^s$, it holds that*

$$\mathop{\mathrm{E}}_{\mathbf{y} \overset{R}{\leftarrow} \mathsf{G}(U_s)} [\mathrm{RealH}_{\mathsf{G}}(\mathbf{y})] = \mathrm{H}(\mathsf{G}(U_s)).$$

Proof. Let $(Y_1, \ldots, Y_m) = \mathsf{G}(U_s)$, and compute

$$\begin{aligned}
\mathop{\mathrm{E}}_{\mathbf{y} \overset{\mathrm{R}}{\leftarrow} \mathsf{G}(U_s)} [\mathrm{RealH}_{\mathsf{G}}(\mathbf{y})] &:= \mathop{\mathrm{E}}_{\mathbf{y} \overset{\mathrm{R}}{\leftarrow} \mathsf{G}(U_s)} \left[\sum_{i \in [m]} \mathrm{H}_{Y_i | Y_{<i}}(\mathbf{y}_i \mid \mathbf{y}_{<i}) \right] \\
&= \sum_{i \in [m]} \mathop{\mathrm{E}}_{\mathbf{y} \overset{\mathrm{R}}{\leftarrow} \mathsf{G}(U_s)} \left[\mathrm{H}_{Y_i | Y_{<i}}(\mathbf{y}_i \mid \mathbf{y}_{<i}) \right] \\
&= \sum_{i \in [m]} \mathrm{H}(Y_i | Y_{<i}) \\
&= \mathrm{H}(\mathsf{G}(U_s)).
\end{aligned}$$

□

4.1.2 Accessible Entropy

Recall that we are interested in *upper bounds* on the accessible entropy of a block generator. We will define two variants of accessible entropy: accessible Shannon entropy and accessible max-entropy. While the Shannon variant is in a sense more intuitive, working with the max-entropy variant, as done in Sections 4.2 and 4.4, yields simpler and more efficient applications. As in the case of real entropy, we connect these two notions through the notion of accessible sample-entropy. For a fixed execution of the adversary $\widetilde{G}$, we ask how surprising were the messages sent by $\widetilde{G}$. We then get accessible Shannon entropy by taking the expectation of this quantity over a random execution and the max-entropy by taking the maximum (up to negligible statistical distance). Here too, the definitions obtained are closely related to the definitions one would obtain by considering a sum of conditional entropies (as we did in the intuitive description earlier). For the Shannon entropy, the definitions would again be identical. (See Lemma 14.)

The definition below differs from the definition of [16], in that we require the bound on the accessible entropy to hold (also) against *nonuniform adversarial generators*. This change simplifies the definitions and proofs, but at the price that we can only construct such inaccessible entropy pseudoentropy generators from functions that are nonuniformly one-way.

Definition 16 (Online block generator). *Let n be a security parameter, and let $m = m(n)$. An m-block* online *generator is a function $\widetilde{G}\colon (\{0,1\}^v)^m \mapsto (\{0,1\}^*)^m$ for some $v = v(n)$, such that the i-th output block of $\widetilde{G}$ is a function of (only) its first i input blocks. We denote the* transcript *of $\widetilde{G}$ over random input by $T_{\widetilde{G}}(1^n) = (R_1, Y_1, \ldots, R_m, Y_m)$, for $(R_1, \ldots, R_m) \overset{R}{\leftarrow} (\{0,1\}^v)^m$ and $(Y_1, \ldots, Y_m) = \widetilde{G}(R_1, \ldots, R_i)$.*

That is, an online block generator is a special type of block generator that tosses fresh random coins before outputting each new block. In the following we let $\widetilde{G}(r_1, \ldots, r_i)_i$ stand for $\widetilde{G}(r_1, \ldots, r_i, x^*)_i$ for arbitrary $x^* \in (\{0,1\}^v)^{m-i}$ (note that the choice of x^* has no effect on the value of $\widetilde{G}(r_1, \ldots, r_i, x^*)_i$).

Definition 17 (Accessible sample-entropy). *Let n be a security parameter, let $m = m(n)$, let $i = i(n) \in [m]$, and let $\widetilde{G}$ be an online m-block online generator. The* accessible sample-entropy of $\mathbf{t} = (r_1, y_1, \ldots, r_i, y_i) \in \mathrm{Supp}(R_1, Y_1 \ldots, R_i, Y_i) = T_{\widetilde{G}}(1^n)_{1,\ldots,2i}$ *is defined as*

$$\mathrm{AccH}_{\widetilde{G}}(\boldsymbol{t}) := \sum_{j \in [i]} \mathrm{AccH}^{j}_{\widetilde{G}}(\boldsymbol{t})$$

for

$$\mathrm{AccH}^{j}_{\widetilde{G}}(\boldsymbol{t}) := \mathrm{H}_{Y_j|R_{<j}}(y_j|r_{<j}).$$

The expected accessible entropy of a random transcript can be expressed in terms of the standard conditional Shannon entropy.

Lemma 14. *Let $\widetilde{G}$ be an online $m = m(n)$-block generator and let $(R_1, Y_1, \ldots, R_m, Y_m) = T_{\widetilde{G}}(1^n)$ be its transcript. Then,*

$$\mathop{\mathrm{E}}_{\boldsymbol{t} \overset{R}{\leftarrow} T_{\widetilde{G}}(1^n)} \left[\sum_{i \in [m]} \mathrm{AccH}^{i}_{\widetilde{G}}(\boldsymbol{t}) \right] = \sum_{i \in [m]} \mathrm{H}(Y_i|R_{<i}).$$

The proof of Lemma 14 is similar to that of Lemma 13.

The above definition is only interesting when putting restrictions on the generator's actions with respect to the underlying generator G. (Otherwise, the accessible entropy of $\widetilde{G}$ can be arbitrarily large by outputting arbitrarily long strings.) In this work, we focus on efficient generators that are consistent with respect to G. That is, the support of their output is contained in that of G.[12]

Definition 18 (Consistent generators). *Let G be a block generator over $\{0,1\}^{s(n)}$. A block (possible online) generator G' over $\{0,1\}^{s'(n)}$ is G consistent if, for every $n \in \mathbb{N}$, it holds that $\mathrm{Supp}(\mathsf{G}'(U_{s'(n)})) \subseteq \mathrm{Supp}(\mathsf{G}(U_{s(n)}))$.*

Definition 19 (Accessible entropy). *A block generator G has accessible entropy at most $k = k(n)$ if, for every efficient, nonuniform, G-consistent, online generator $\widetilde{G}$ and all large enough n,*

$$\mathop{\mathrm{E}}_{\boldsymbol{t} \overset{R}{\leftarrow} T_{\widetilde{G}}(1^n)} \left[\mathrm{AccH}_{\widetilde{G}}(\boldsymbol{t}) \right] \leq k.$$

The generator G has accessible max-entropy at most k if

$$\Pr_{\boldsymbol{t} \overset{R}{\leftarrow} T_{\widetilde{G}}(1^n)}[\mathrm{AccH}_{\widetilde{G}}(\boldsymbol{t}) > k] = \mathrm{neg}(n),$$

for every such $\widetilde{G}$.

In Section 4.2, we prove the existence of one-way functions implies that of an inaccessible max-entropy entropy generator: an efficient block generator whose accessible entropy is noticeably larger than its accessible entropy. The converse direction is also true.

Lemma 15. *Let G be an efficient block generator with real entropy $k(n)$, and assume that G has accessible entropy, or accessible max-entropy, at most $k(n) - 1/p(n)$, for some $p \in \mathrm{poly}$. Then one-way functions exist.*[13]

Proof. Omitted. □

[12] In the more complicated notion of accessible entropy considered in [11], the "generator" needs to *prove* that its output blocks are in the support of G, by providing an input of G that would have generated the same blocks. It is also allowed there for a generator to fail to prove the latter with some probability, which requires a measure of accessible entropy that discounts entropy that may come from failing.

[13] Specifically, one can show that a variant of $f(x,i) = \mathsf{G}(x)_{1,\ldots,i}$ is a "distributional" one-way function.

4.2 Inaccessible Entropy Generator from One-way Functions

In this section, we show how to build an inaccessible entropy generator from any one-way function. In particular, we prove the following theorem:

Construction 9. *For $f\colon \{0,1\}^n \mapsto \{0,1\}^n$, define the $(n+1)$ block generator G^f over $\{0,1\}^n$ by*

$$\mathsf{G}^f(x) = (f(x)_1, \ldots, f(x)_n, x)$$

Namely, the first n blocks of $\mathsf{G}^f(x)$ are the bits of $f(x)$, and its final block is x.

Theorem 10 (Inaccessible entropy generators from one-way functions). *If $f\colon \{0,1\}^n \mapsto \{0,1\}^n$ is nonuniformly one-way, then the efficient block generator $\mathsf{G} = \mathsf{G}^f$ defined in Construction 9 has accessible max-entropy at most $n - \omega(\log n)$.*

Remark 5 (Tighter reduction). [16] prove an analog theorem for the $O(n/\log n)$-block generator that groups each consecutive $\log n$ bits of $f(n)$ into a single block.

Proof. Suppose on the contrary that there exists an efficient, nonuniform, G-consistent online block generator $\widetilde{G}$ such that

$$\Pr_{\mathbf{t} \overset{R}{\leftarrow} T_{\widetilde{G}}(1^n)} \left[\mathrm{AccH}_{\widetilde{G}}(\mathbf{t}) > n - c \cdot \log n\right] > \varepsilon(n) \tag{.15}$$

for some constant $c > 0$, $\varepsilon(n) = 1/\operatorname{poly}(n)$, and infinitely many n's. In the following we fix $n \in \mathbb{N}$ for which the above equation holds, and omit it from the notation when its value is clear from the context. Let $m = n + 1$ and let v be abound on the number of bits used by $\widetilde{G}$ in each round. The inverter Inv for f is defined as follows:

Algorithm 11 (Inverter Inv for f from the accessible entropy generator $\widetilde{G}$)**.**

Input: $z \in \{0,1\}^n$
Operation:

```
1. For $i = 1$ to $n$,
    a. Sample $r_i \overset{R}{\leftarrow} \{0,1\}^v$ and let $y_i = \widetilde{G}(r_1, \ldots, r_i)_i$.
    b. If $y_i = z_i$, move to next value of $i$.
    c. Abort after $n^2/\varepsilon$ failed attempts for sampling good $r_i$.
2. Sample $r_m \overset{R}{\leftarrow} \{0,1\}^v$ and output $\widetilde{G}(r_1, \ldots, r_m)_m$.
```

Namely, $\mathrm{Inv}(y)$ does the only natural thing one can do with $\widetilde{G}$; it tries to make, via rewinding, $\widetilde{G}$'s first n output blocks equal to y, knowing that, if this happens then since $\widetilde{G}$ is G-consistent, $\widetilde{G}$'s m-th output block is a preimage of y.

It is clear that Inv runs in polynomial time, so we will finish the proof by showing that

$$\Pr_{y \overset{R}{\leftarrow} f(U_n)} \left[\mathrm{Inv}(y) \in f^{-1}(y)\right] \geq \varepsilon^2/16n.$$

We prove the above by relating the transcript distribution induced by the standalone execution of $\widetilde{G}(1^n)$ to that induced by the embedded execution of $\widetilde{G}$ in $\mathrm{Inv}(f(U_n))$. In more detail, we show that high-accessible-entropy transcripts with respect to the standalone execution of G, i.e., $\mathrm{AccH}_{\widetilde{G}}(\mathbf{t}) > n - c \cdot \log n$, happen with not much smaller probability also in the emulated execution. Since whenever Inv does not abort it is guaranteed to invert y, it follows that the success probability of Inv is lower bounded by the probability that $\widetilde{G}(1^n)$ outputs a high-accessible-entropy transcript, and thus is nonnegligible.

For intuition about why the above statement about high-accessible-entropy transcripts is true, consider the case of a one-way *permutation* f. By definition, high-accessible-entropy transcripts in the stand alone execution of $\widetilde{G}$ happen with probability at most $\mathrm{poly}(n)/2^n$. On the other hand, the probability that a "typical" transcript is produced by the emulated execution of $\widetilde{G}$ is about 2^{-n}—the probability that random output of f equals the transcript's first n output blocks.

Proving the above formally for arbitrary one-way functions is the subject of the following proof:

Standalone execution $\widetilde{G}(1^n)$. Let $\widetilde{T} = T_{\widetilde{G}}$, and recall that $\widetilde{T} = (\widetilde{R}_1, \widetilde{Y}_1, \ldots, \widetilde{R}_m, \widetilde{Y}_m)$ is associated with a random execution of $\widetilde{G}$ on security parameter n by

- $\widetilde{R}_i$ – the random coins of $\widetilde{G}$ in the i-th round, and
- $\widetilde{Y}_i$ – $\widetilde{G}$'s i-th output block.

Recall that, for $\mathbf{t} = (r_1, y_1, \ldots, r_m, y_m) \in \mathrm{Supp}(\widetilde{T})$, we have defined

$$\mathrm{AccH}_{\widetilde{G}}(\mathbf{t}) := \sum_{i \in [m]} \mathrm{H}_{Y_j|R_{<j}}(y_j|r_{<j}).$$

Compute

$$\begin{aligned} \Pr_{\widetilde{T}}[\mathbf{t}] &= \prod_{i=1}^{m} \Pr_{\widetilde{Y}_i|\widetilde{R}_{<i}}[y_i|r_{<i}] \cdot \Pr_{\widetilde{R}_i|\widetilde{R}_{<i},\widetilde{Y}_i}[r_i|r_{<i}, y_i] \\ &= 2^{-\sum_{i=1}^{m} \mathrm{H}_{\widetilde{Y}_i|\widetilde{R}_{<i}}(y_i|r_{<i})} \cdot \prod_{i=1}^{m} \Pr_{\widetilde{R}_i|\widetilde{R}_{<i},\widetilde{Y}_i}[r_i|r_{<i}, y_i] \\ &= 2^{-\mathrm{AccH}_{\widetilde{G}}(\mathbf{t})} \cdot R(\mathbf{t}) \end{aligned} \tag{.16}$$

for

$$R(\mathbf{t}) := \prod_{i=1}^{m} \Pr_{\widetilde{R}_i|\widetilde{R}_{<i},\widetilde{Y}_i}[r_i|r_{<i}, y_i]. \tag{.17}$$

Execution embedded in $\mathrm{Inv}_{\mathrm{g}}(f(U_n))$. Let $\widehat{T} = (\widehat{R}_1, \widehat{Y}_1, \ldots, \widehat{R}_m, \widehat{Y}_m)$ denote the value of $\widetilde{G}$'s coins and output blocks, of the execution done in step 2 of a random execution of the *unbounded* version of Inv (i.e., step 1.(c) is removed) on input $Z = (Z_1, \ldots, Z_{m-1}) = f(U_n)$. (This unboundedness change is only an intermediate step in the proof that does not significantly change the inversion probability of Inv, as shown below.)

Since $\widetilde{G}$ is G-consistent, it holds that $(y_1, \ldots, y_{m-1}) \in \mathrm{Supp}(f(U_n))$ for every $(r_1, y_1, \ldots, r_m, y_m) \in \mathrm{Supp}(\widetilde{T})$. It follows that every $\mathbf{t} \in \mathrm{Supp}(\widetilde{T})$ can be "produced" by the unbounded version of Inv, and therefore $\mathrm{Supp}(\widetilde{T}) \subseteq \mathrm{Supp}(\widehat{T})$. For $\mathbf{t} \in \mathrm{Supp}(\widetilde{T})$, we compute

$$
\begin{aligned}
\Pr_{\widehat{T}}[\mathbf{t}] &= \prod_{i=1}^{m} \Pr_{\widehat{Y}_i|\widehat{R}_{<i}}[y_i|r_{<i}] \cdot \Pr_{\widehat{R}_i|\widehat{R}_{<i},\widehat{Y}_i}[r_i|r_{<i},y_i] \qquad (.18)\\
&= \left(\prod_{i=1}^{m-1} \Pr_{Z_i|\widehat{Y}_{<i}}[y_i|y_{<i}] \cdot \Pr_{\widehat{Y}_i|\widehat{R}_{<i},Z_i}[y_i|r_{<i},y_i]\right) \cdot \Pr_{\widehat{Y}_m|\widehat{R}_{<m}}[y_m|r_{<m}] \cdot \prod_{i=1}^{m} \Pr_{\widehat{R}_i|\widehat{R}_{<i},\widehat{Y}_i}[r_i|r_{<i},y_i]\\
&= \left(\prod_{i=1}^{m-1} \Pr_{Z_i|\widehat{Y}_{<i}}[y_i|y_{<i}] \cdot 1\right) \cdot \Pr_{\widehat{Y}_m|\widehat{R}_{<m}}[y_m|r_{<m}] \cdot \prod_{i=1}^{m} \Pr_{\widehat{R}_i|\widehat{R}_{<i},\widehat{Y}_i}[r_i|r_{<i},y_i]\\
&= \Pr_{f(U_n)}[y_{<m}] \cdot \Pr_{\widehat{Y}_m|\widehat{R}_{<m}}[y_m|r_{<m}] \cdot R(\mathbf{t})\\
&= \Pr_{f(U_n)}[y_{<m}] \cdot \Pr_{\widetilde{Y}_m|\widetilde{R}_{<m}}[y_m|r_{<m}] \cdot R(\mathbf{t}).
\end{aligned}
$$

Note that in the last line we moved from conditioning on $\widehat{R}_{<m}$ to conditioning on $\widetilde{R}_{<m}$. The third equality holds since $\mathbf{t} \in \mathrm{Supp}(\widetilde{T})$ and Inv is unbounded.

Relating the two distributions. Combining Equations (.16) and (.18) yields that, for $\mathbf{t} = (r_1, y_1, \ldots, r_m, y_m) \in \mathrm{Supp}(\widetilde{T})$, it holds that

$$
\Pr_{\widehat{T}}[\mathbf{t}] = \Pr_{\widetilde{T}}[\mathbf{t}] \cdot \left(\Pr_{f(U_n)}[y_{<m}] \cdot \Pr_{\widetilde{Y}_m|\widetilde{R}_{<m}}[y_m|r_{<m}] \cdot 2^{\mathrm{AccH}_{\widetilde{G}}(\mathbf{t})}\right). \qquad (.19)
$$

In particular, if $\mathrm{AccH}_{\widetilde{G}}(\mathbf{t}) \geq n - c\log n$, then

$$
\begin{aligned}
\Pr_{\widehat{T}}[\mathbf{t}] &\geq \Pr_{\widetilde{T}}[\mathbf{t}] \cdot \frac{2^n \cdot \Pr_{f(U_n)}[y_{<m}]}{n^c} \cdot \Pr_{\widetilde{Y}_m|\widetilde{R}_{<m}}[y_m|r_{<m}] \qquad (.20)\\
&= \Pr_{\widetilde{T}}[\mathbf{t}] \cdot \frac{\left|f^{-1}(y_{<m})\right|}{n^c} \cdot \Pr_{\widetilde{Y}_m|\widetilde{R}_{<m}}[y_m|r_{<m}].
\end{aligned}
$$

If it is also the case that $\mathrm{H}_{\widetilde{Y}_m|\widetilde{R}_{<m}}(y_m|r_{<m}) \leq \log\left|f^{-1}(y_{<m})\right| + k$ for some $k > 0$, then

$$
\Pr_{\widehat{T}}[\mathbf{t}] \geq \Pr_{\widetilde{T}}[\mathbf{t}] \cdot \frac{\left|f^{-1}(y_{<m})\right|}{n^c} \cdot \frac{2^{-k}}{\left|f^{-1}(y_{<m})\right|} = \frac{\Pr_{\widetilde{T}}[\mathbf{t}]}{2^k n^c} \qquad (.21)
$$

Lower bounding the inversion probability of Inv. We conclude the proof by showing that Equation (.21) implies the existence of a large set of transcripts that (the bounded version of) Inv performs well upon.

Let $\mathcal{S}$ denote the set of transcripts $\mathbf{t} = (r_1, y_1, \ldots, r_m, y_m) \in \mathrm{Supp}(\widetilde{T})$ with

1. $\mathrm{AccH}_{\widetilde{G}}(\mathbf{t}) \geq n - c\log n$,
2. $\mathrm{H}_{\widetilde{Y}_m|\widetilde{R}_{<m}}(y_m|r_{<m}) \leq \log\left|f^{-1}(y_{<m})\right| + \log(4/\varepsilon)$, and
3. $\mathrm{H}_{\widetilde{Y}_i|\widetilde{Y}_{<i}}(y_i|y_{<i}) \leq \log(4n/\varepsilon)$ for all $i \in [m-1]$.

The first two properties will allow us to use Equations (.20) and (.21) to argue that, if $\mathcal{S}$ happens with significant probability with respect to $\widetilde{T}$, then this holds also with respect to $\widehat{T}$. The last property will allow us to show that this happens also with respect to the bounded version of Inv. We start by showing that $\mathcal{S}$ happens with significant probability with respect to $\widetilde{T}$, then show that this holds also with respect to $\widehat{T}$, and finally use it to lowerbound the success probability of Inv.

By Lemma 1,

$$
\Pr_{(r_1,y_1,\ldots,r_m,y_m)\xleftarrow{\mathrm{R}}\widetilde{T}}\left[\mathrm{H}_{\widetilde{Y}_m|\widetilde{R}_{<m}}(y_m|r_{<m}) > \log\left|f^{-1}(y_{<m})\right| + k\right] < 2^{-k} \qquad (.22)
$$

for any $k > 0$, where since $\left|\mathrm{Supp}(\widetilde{Y}_i)\right| = 1$ for all $i \in [m-1]$, it holds that

$$\Pr_{(y_1,\ldots,y_m)\overset{\mathrm{R}}{\leftarrow}(\widetilde{Y}_1,\ldots,\widetilde{Y}_m)}\left[\exists i\in[m-1]\colon\ \mathrm{H}_{\widetilde{Y}_i|\widetilde{Y}_{<i}}(y_i|y_{<i})>v\right]<(m-1)\cdot 2^{-v} \tag{.23}$$

for any $v>0$.

Applying Equations (.22) and (.23) with $k=\log(4/\varepsilon)$ and $v=\log(4n/\varepsilon)$, respectively, and recalling that, by assumption, $\Pr_{\mathbf{t}\overset{\mathrm{R}}{\leftarrow}\widetilde{T}}\left[\mathrm{AccH}_{\widetilde{G}}(\mathbf{t})\geq n-c\log n\right]\geq\varepsilon$, yields that

$$\Pr_{\widetilde{T}}[\mathcal{S}]\geq\varepsilon-\frac{\varepsilon}{4}-\frac{\varepsilon}{4}=\varepsilon/2 \tag{.24}$$

By Equation (.21) and the first two properties of $\mathcal{S}$, we have that

$$\Pr_{\widehat{T}}[\mathcal{S}]\geq\frac{\varepsilon}{4n^c}\cdot\Pr_{\widetilde{T}}[\mathcal{S}]\geq\frac{\varepsilon^2}{8n^c}. \tag{.25}$$

Finally, let $\widehat{T}'$ denote the final value of $\widetilde{G}$'s coins and output blocks, induced by the *bounded* version of Inv (set to $\perp$ if Inv aborts). The third property of $\mathcal{S}$ yields that

$$\Pr_{\widehat{T}'}[\mathbf{t}]\geq\Pr_{\widehat{T}}[\mathbf{t}]\cdot\left(1-(m-1)\cdot(1-\tfrac{\varepsilon}{4n})^{n^2/\varepsilon}\right)\geq\Pr_{\widehat{T}}[\mathbf{t}]\cdot(1-O(m\cdot 2^{-n}))\geq\Pr_{\widehat{T}}[\mathbf{t}]/2 \tag{.26}$$

for every $\mathbf{t}\in\mathcal{S}$. We conclude that

$$\begin{aligned}\Pr_{z\overset{\mathrm{R}}{\leftarrow}f(U_n)}\left[\mathrm{Inv}(z)\in f^{-1}(z)\right]&=\Pr_{z\overset{\mathrm{R}}{\leftarrow}f(U_n)}[\mathrm{Inv}(z)\text{ does not abort}]\\&\geq\Pr_{\widehat{T}'}[\mathcal{S}]\\&\geq\frac{1}{2}\cdot\Pr_{\widehat{T}}[\mathcal{S}]\\&\geq\frac{\varepsilon^2}{16n^c}.\end{aligned}$$

□

4.3 Manipulating Real and Accessible Entropy

Following are two tools to manipulate the real and accessible entropy of a block generator. Since we are dealing with the more complex accessible entropy notion, the statements and proofs of the following lemmas are more complicated than those of Section 3.3 (given for the next-block entropy notion). Yet, the bottom line of the lemmas is essentially the same.

4.3.1 Entropy Equalization via Truncated Sequential Repetition

Similarly to what happens in Section 3, this tool concatenates several independent executions of an m-block generator, and then truncates, at random, some of the first and final output blocks of the concatenated string. Assuming that the (overall) real entropy of the original generator is at least k_{REAL}, then the real entropy of each block of the resulting generator is at least k_{REAL}/m. This per-block knowledge of the real entropy, is very handy when considering applications of inaccessible entropy generators, and in particular for constructions of statistically hiding commitment.

The price of this manipulation is that we "give away" some real entropy (as we do not output all blocks), while we cannot claim that the same happens to the accessible entropy. Hence, the additive gap between the real and accessible entropy of the resulting generator gets smaller. Yet, if we do enough repetition, this loss is insignificant.

Definition 20. *For security parameter n, let $m = m(n)$, let $s = s(n)$, $w = w(n)$, and let $s' = s'(n) = \log(m(n)) + w(n) \cdot s(n)$. Given an m-block generator G over $\{0,1\}^s$, define the $((w-1)\cdot m)$-block generator $\mathsf{G}^{[w]}$ over $[m] \times (\{0,1\}^s)^w$ as follows: on input $(j,(x_1,\ldots,x_w)) \in [m] \times (\{0,1\}^s)^w$, it sets $\mathbf{y} = (y_1,\ldots,y_{wm}) = (\mathsf{G}(x_1),\ldots,\mathsf{G}(x_w))$, and outputs $((j,y_j),y_{j_1},\ldots,y_{(w-1)m+j-1})$.*

That is, $\mathsf{G}^{[w]}$ truncates the first $j-1$ and last $m+1-j$ blocks of $\mathbf{y}$, and outputs the remaining $(w-1)\cdot m$ blocks one by one, while appending j to each block it outputs. (Using the terminology of Section 3, $\mathsf{G}^{[w]}$ outputs Equalizer$^m(j,y_1,\ldots,y_{wm})$, where Equalizer is according to Definition 12, while appending j to the first block.)

Lemma 16. *For security parameter n, let $m = m(n)$ be a power of 2, let $s = s(n)$ and let G be an efficient m-block generator over $\{0,1\}^s$, and let $w = w(n)$ be a polynomially computable and bounded integer function. Then, $\mathsf{G}^{[w]}$ defined according to Definition 20 is an efficient,[14] $((w-1)\cdot m)$-block generator that satisfies the following properties:*

Real entropy: *If G has real entropy at least $k_{\text{REAL}} = k_{\text{REAL}}(n)$, then each block of $\mathsf{G}^{[w]}$ has real entropy at least k_{REAL}/m.*

Accessible max-entropy: *The following holds for any $d = d(n) \in \omega(\log n)$. If G has accessible max-entropy at most $k_{\text{ACC}} = k_{\text{ACC}}(n)$, then $\mathsf{G}^{[w]}$ has accessible max-entropy at most*

$$k'_{\text{ACC}} := (w-2)\cdot k_{\text{ACC}} + 2 \cdot \mathrm{H}_0(\mathsf{G}(U_s)) + \log(m) + d.$$

Roughly, each of the $(w-2)$ non truncated executions of G embedded in $\mathsf{G}^{[w]}$ contributes its accessible entropy to the overall accessible entropy of $\mathsf{G}^{[w]}$. In addition, we pay the max-entropy of the two truncated executions of G embedded in $\mathsf{G}^{[w]}$.

Proof. To avoid notational clutter let $\mathbb{G} = \mathsf{G}^{[w]}$.

Real entropy. The proof of this part is very similar to the proof of the first part of Lemma 10. Fix $n \in \mathbb{N}$ and omit it from the notation when clear from the context. Let $\tilde{m} = (w-1)m$, let $\widetilde{Y} = \mathbb{G}(U_{s'} = (J, X_1,\ldots,X_w))$, let $Y^{(w)} = (\mathsf{G}(X_1),\ldots,\mathsf{G}(X_w))$, and finally for $i \in [wm]$, let ${Y^{(w)}}'_i = (J, Y^{(w)}_i)$ if $J = i$, and $Y^{(w)}_i$ otherwise. For $i \in [\tilde{m}]$, compute

$$\begin{aligned}\mathrm{H}(\widetilde{Y}_i \mid \widetilde{Y}_{<i}) &= \mathrm{H}({Y^{(w)}}'_{i+J-1} \mid {Y^{(w)}}'_{J,\ldots,i+J-2}) \\ &\geq \mathrm{H}(Y^{(w)}_{i+J-1} \mid Y^{(w)}_{J,\ldots,i+J-2}, J).\end{aligned}$$

The proof continues as the first part of the proof of Lemma 10.

Accessible entropy. To establish the statement on the accessible entropy, let $\widetilde{\mathbb{G}}$ be an efficient $\mathbb{G}$-consistent generator, and let

$$\varepsilon = \varepsilon(n) := \Pr_{\mathbf{t} \xleftarrow{R} \widetilde{\mathbb{T}}}\left[\mathrm{AccH}_{\widetilde{\mathbb{G}}}(\mathbf{t}) > k'_{\text{ACC}}\right] \tag{.27}$$

for $\widetilde{\mathbb{T}} = T_{\widetilde{\mathbb{G}}}(1^n)$. Our goal is to show that ε is negligible in n. We do that by finding a subtranscript of $\widetilde{\mathbb{T}}$ that, with high probability, contributes more than k_{ACC} bits of accessible entropy, if the overall accessible entropy of $\widetilde{\mathbb{T}}$ is more than k'_{ACC}. We then use this observation to construct a cheating generator for G that achieves accessible entropy greater than k_{ACC} with probability that is negligibly close to ε.

Let $(R_1, Y_1, \ldots, R_{\tilde{m}}, Y_{\tilde{m}}) = \widetilde{\mathbb{T}}$ and let J be the first part of Y_1 (recall that Y_1 is of the form $(j,\cdot)$). Fix $j \in [m]$, and let $(R^j_1, Y^j_1, \ldots, R^j_{\tilde{m}}, Y^j_{\tilde{m}}) = \widetilde{\mathbb{T}}^j = \widetilde{\mathbb{T}}|_{J=j}$. Let $\mathcal{I} = \mathcal{I}(j)$ be the indices of the blocks coming

[14] Since m is a power of 2, changing the input domain of $\mathsf{G}^{[w]}$ to $\{0,1\}^{s'}$ for some polynomial-bounded and polynomial-time computable s', to make it an efficient block generator according to Definition 13, can be done by standard techniques.

from the truncated executions of G in $\mathbb{G}$ (i.e., $\{1,\ldots,m+1-j\} \cup \{\tilde{m}+2-j,\ldots,\tilde{m}\}$). Our first step is to show that these blocks do not contribute much more entropy than the max-entropy of $\mathsf{G}(U_n)$. Specifically, by Lemma 7, letting $\mathbf{X} = (Y_1^j, R_1^j, \ldots, Y_{\tilde{m}}^j, R_{\tilde{m}}^j)$ and $\mathcal{J} = \mathcal{I}$, it holds that

$$\Pr_{\mathbf{t}=(r_1,y_1,\ldots,r_{\tilde{m}},y_{\tilde{m}}) \xleftarrow{\mathrm{R}} \widetilde{\mathbb{T}}^j}\left[\sum_{i\in\mathcal{I}} \mathrm{H}_{Y_i^j|R_{<i}^j}(y_i|r_{<i}) > 2\cdot \mathrm{H}_0(\mathsf{G}(U_s)) + d/2\right] \leq 2\cdot 2^{-d/2} = \mathrm{neg}(n). \tag{.28}$$

Namely, with save but negligible probability, the blocks that relate to the truncated executions of G in $\mathbb{G}$, do not contribute much more than their support size to the overall accessible entropy.

Our next step is to remove the conditioning on $J = j$ (that we have introduced to have the indices of interest fixed, which enabled us to use Lemma 7). By Lemma 1, it holds that

$$\Pr_{j \xleftarrow{\mathrm{R}} J}\left[\mathrm{H}_J(j) > \log(m) + d/2\right] \leq 2^{-d/2} = \mathrm{neg}(n). \tag{.29}$$

Since for every $i > 1$ and $(r_1, y_1 = (j,\cdot), \ldots, r_{\tilde{m}}, y_{\tilde{m}}) \in \mathrm{Supp}(\widetilde{\mathbb{T}})$, it holds that $\mathrm{H}_{Y_i|R_{<i}}(y_i|r_{<i}) = \mathrm{H}_{Y_i^j|R_{<i}^j}(y_i|r_{<i})$, and for $i = 1$, it holds that $\mathrm{H}_{Y_1}(y_1) = \mathrm{H}_J(j) + \mathrm{H}_{Y_1^j}(y_1)$, the above yields that

$$\Pr_{\mathbf{t}=(r_1,y_1,\ldots,r_{\tilde{m}},y_{\tilde{m}}) \xleftarrow{\mathrm{R}} \widetilde{\mathbb{T}}}\left[\sum_{i\in[\tilde{m}]\setminus\mathcal{I}(J)} \mathrm{H}_{Y_i|R_{<i}}(y_i|r_{<i}) > (w-2)\cdot k_{\mathrm{ACC}}\right] \geq \varepsilon - \mathrm{neg}(n). \tag{.30}$$

Let $\mathcal{F}(j) = \{km + 2 - j \colon k \in [w-2]\}$, i.e., the indices of the first blocks of the non-truncated executions of G in $\mathbb{G}$, when the first block of $\mathbb{G}$ is $(j,\cdot)$. It follows that,

$$\Pr_{\mathbf{t}=(r_1,y_1,\ldots,r_{\tilde{m}},y_{\tilde{m}}) \xleftarrow{\mathrm{R}} \widetilde{\mathbb{T}}}\left[\exists f \in \mathcal{F}(J) \colon \sum_{i=f}^{f+m-1} \mathrm{H}_{Y_i|R_{<i}}(y_i|r_{<i}) > k_{\mathrm{ACC}}\right] \geq \varepsilon - \mathrm{neg} \tag{.31}$$

In particular, there exist $j^* \in [m]$, $f^* \in \mathcal{F}(j^*)$, and $\mathbf{r}^* \in \mathrm{Supp}(R_{<f^*}|_{J=j^*})$ such that

$$\Pr_{\mathbf{t}=(r_1,y_1,\ldots,r_{\tilde{m}},y_{\tilde{m}}) \xleftarrow{\mathrm{R}} \widetilde{\mathbb{T}}}\left[\sum_{i=f^*}^{f^*+m-1} \mathrm{H}_{Y_i|R_{<i}}(y_i|r_{<i}) > k_{\mathrm{ACC}} \mid r_{<f^*} = \mathbf{r}^*\right] \geq (\varepsilon - \mathrm{neg})/m. \tag{.32}$$

Consider the efficient, nonuniform, G-consistent generator $\widetilde{G}$ that acts as follows: it starts a random execution of $\widetilde{\mathbb{G}}$ with its first (f^*-1) randomness blocks fixed to $\mathbf{r}^*$, and outputs the blocks indexed by $\{f^*,\ldots,f^*+m-1\}$. Let $(R'_1, Y'_1, \ldots, R'_m, Y'_m) = T_{\widetilde{G}}$ be the transcript of $\widetilde{G}$. It is easy to verify that, for every $(r_1, y_1, \ldots, r_m, y_m) \in \mathrm{Supp}(T_{\widetilde{G}})$ and $1 < i \leq m$, it holds that

$$\mathrm{H}_{Y'_i|R'_{<i}}(y_i|r_{<i}) = \mathrm{H}_{Y_{f+i}|R_{<f+i}}(y_i|(\mathbf{r}^*, r_{<i})). \tag{.33}$$

Thus, Equation (.32) yields that

$$\Pr_{\mathbf{t} \xleftarrow{\mathrm{R}} T_{\widetilde{G}}}\left[\mathrm{AccH}_{\widetilde{G}}(\mathbf{t}) > k_{\mathrm{ACC}}\right] > (\varepsilon - \mathrm{neg}(n))/m.$$

Hence, the assumption about the inaccessible entropy of G yields that ε is a negligible function of n, and the proof of the lemma follows. □

4.3.2 Parallel Repetition

This manipulation simply takes parallel repetition of a generator. The effect of this manipulation is twofold. The first effect is that the overall real entropy of a v-fold parallel repetition of a generator G is v times the real entropy of G. Hence, if G real entropy is larger than its accessible entropy, this gap get multiplied by v in the resulting variable. The second effect of such repetition is turning per-block real entropy into per-block *min-entropy*. The price of this manipulation is a slight decrease in the per block min-entropy of the resulting generator, compared to the sum of the per block real entropies of the independent copies of the generators used to generate it. (This loss is due to the move from Shannon entropy to min-entropy, rather than from the parallel repetition itself.) But when taking enough copies, this loss can be ignored.

Definition 21. *Let $m = m(n)$, $s = s(n)$, and $v = v(n)$. Given an m-block generator G over $\{0,1\}^s$, define the m-block generator $G^{\langle v\rangle}$ over $(\{0,1\}^s)^v$ as follows: on input $(x_1,\ldots,x_v) \in (\{0,1\}^s)^v$, the i-th block of $G^{\langle v\rangle}$ is $(\mathsf{G}(x_1)_i,\ldots,\mathsf{G}(x_v)_i)$.*

Lemma 17. *For security parameter n, let $m = m(n)$, let $v = v(n)$ be polynomial-time polynomially computable and bounded integer functions, and let G be an efficient,[15] m-block generator. Then $G^{\langle v\rangle}$, defined according to Definition 21, is an efficient m-block generator that satisfies the following properties:*

Real entropy: If each block of G has real min-entropy at least $k_{\text{REAL}} = k_{\text{REAL}}(n)$, then each block of $G^{\langle v\rangle}$ has real min-entropy at least $k'_{\text{REAL}}(n) = v \cdot k_{\text{REAL}} - O((\log n + \ell)\cdot \log n \cdot \sqrt{v})$, for $\ell = \ell(n)$ being the maximal block length of G.

Accessible max-entropy: The following holds for every $d = d(n) \in \omega(\log n)$. If G has accessible max-entropy at most $k_{\text{ACC}} = k_{\text{ACC}}(n)$, then $G^{\langle v\rangle}$ has accessible max-entropy at most $k'_{\text{ACC}}(n) = v\cdot k_{\text{ACC}} + d\cdot m$.

Proof. The bound on real entropy follows readily from Lemma 5 by taking $\varepsilon = 2^{-\log^2 n}$, and noting that the support size of each block of G is at most $\ell \cdot 2^\ell$. Therefore, we focus on establishing the bound on accessible max-entropy. Let $\mathbb{G} = G^{\langle v\rangle}$, let $\widetilde{\mathbb{G}}$ be an efficient, nonuniform, $\mathbb{G}$-consistent generator, and let

$$\varepsilon = \varepsilon(n) := \Pr_{\mathbf{t}\xleftarrow{R}\widetilde{\mathbb{T}}}\left[\mathrm{AccH}_{\widetilde{\mathbb{G}}}(\mathbf{t}) > k'_{\text{ACC}}\right] \tag{.34}$$

for $\widetilde{\mathbb{T}} = T_{\widetilde{\mathbb{G}}}(1^n)$. Our goal is to show that ε is negligible in n.

Let $(R_1, Y_1, \ldots, R_m, Y_m) = \widetilde{\mathbb{T}}$. By definition, for $\mathbf{t} = (r_1, y_1, \ldots, r_m, y_m) \in \mathrm{Supp}(\widetilde{\mathbb{T}})$,

$$\mathrm{AccH}_{\widetilde{\mathbb{G}}}(\mathbf{t}) = \sum_{i\in[m]} \mathrm{H}_{Y_i|R_{<i}}(y_i \mid r_{<i}). \tag{.35}$$

Since $\widetilde{\mathbb{G}}$ is $\mathbb{G}$-consistent, each Y_i is of the form $(Y_{i,1},\ldots,Y_{i,v})$. Lemma 6, taking $\mathbf{X} = Y_i|_{R_{<i}=r_{<i}}$, yields that

$$\Pr_{\mathbf{t}=(r_1,y_1,\ldots,r_m,y_m)\xleftarrow{R}\widetilde{\mathbb{T}}}\left[\mathrm{H}_{Y_i|R_{<i}}(y_i|r_{<i}) > d + \sum_{j=1}^{v}\mathrm{H}_{Y_{i,j}|R_{<i}}(y_{i,j}|r_{<i})\right] \le 2^{-d} = \mathrm{neg}(n) \tag{.36}$$

for every $i \in [m]$. Summing over all $i \in [m]$, we get that

$$\Pr_{\mathbf{t}=(r_1,y_1,\ldots,r_m,y_m)\xleftarrow{R}\widetilde{\mathbb{T}}}\left[\sum_{i\in[m]}\mathrm{H}_{Y_i|R_{<i}}(y_i|r_{<i}) > md + \sum_{i\in[m]}\sum_{j\in[v]}\mathrm{H}_{Y_{i,j}|R_{<i}}(y_{i,j}|r_{<i})\right] = \mathrm{neg}(n) \tag{.37}$$

and therefore

[15] Changing the input domain of G to $\{0,1\}^{s'(n)}$ for some polynomial-bounded and polynomial-time computable s', to make it an efficient block generator according to Definition 13, can be done by standard techniques.

$$\Pr_{\mathbf{t}=(r_1,y_1,\ldots,r_m,y_m)\xleftarrow{R}\widetilde{\mathbb{T}}}\left[\sum_{i\in[m]}\sum_{j\in[v]} \mathrm{H}_{Y_{i,j}|R_{<i}}(y_{i,j}|r_{<i}) \geq k'_{\mathrm{ACC}} - m\cdot d\right] \geq \varepsilon - \mathrm{neg}(n). \tag{.38}$$

In particular, there exist $j^* \in [v]$ such that

$$\Pr_{\mathbf{t}=(r_1,y_1,\ldots,r_m,y_m)\xleftarrow{R}\widetilde{\mathbb{T}}}\left[\sum_{i\in[m]} \mathrm{H}_{Y_{i,j^*}|R_{<i}}(y_{i,j^*}|r_{<i}) > k_{\mathrm{ACC}} = (k'_{\mathrm{ACC}} - m\cdot d)/v\right] \geq \varepsilon - \mathrm{neg}(n) \tag{.39}$$

Consider the following efficient, nonuniform, G-consistent generator $\widetilde{G}$. This generator starts a random execution of $\widetilde{\mathbb{G}}$, and outputs y_{i,j^*} as its i-th block, for $y_i = (y_{i,1},\ldots,y_{i,v})$ being the i-th block (locally) output by $\widetilde{\mathbb{G}}$. Let $(R'_1,Y'_1,\ldots,R'_m,Y'_m) = T_{\widetilde{G}}$. It is easy to verify that, for every $(r_1,y_1,\ldots,r_m,y_m) \in \mathrm{Supp}(T_{\widetilde{G}})$ and $1 < i \leq m$, it holds that

$$\mathrm{H}_{Y'_i|R'_{<i}}(y_i|j,r_{<i}) = \mathrm{H}_{Y_{i,j^*}|R_{<i}}(y_i|r_{<i}). \tag{.40}$$

Thus, Equation (.39) yields that

$$\Pr_{\mathbf{t}\xleftarrow{R}T_{\widetilde{G}}}\left[\mathrm{AccH}_{\widetilde{G}}(\mathbf{t}) > k_{\mathrm{ACC}}\right] \geq \varepsilon - \mathrm{neg}(n).$$

The assumption about the inaccessible entropy of G yields that ε is negligible in n, and the proof of the lemma follows. □

4.4 Inaccessible Entropy Generator to Statistically Hiding Commitment

In this section we prove a simplified version of the construction of statistically hiding commitments from inaccessible entropy generators. Specifically, we only prove a weaker version of Lemma 18, stated below, which is the main lemma in this reduction. But first, we recall the definition of such commitment schemes.

Statistically hiding commitment schemes. A *commitment scheme* is the cryptographic analogue of a safe. It is a two-party protocol between a *sender* S and a *receiver* R that consists of two stages. The *commit stage* corresponds to putting an object in a safe and locking it; the sender "commits" to a private message m. The *reveal stage* corresponds to unlocking and opening the safe; the sender "reveals" the message m and "proves" that it was the value committed to in the commit stage (without loss of generality by revealing coin tosses consistent with m and the transcript of the commit stage).

Definition 22. *A* (bit) commitment scheme[16] *is an efficient two-party protocol* $\mathsf{Com} = (\mathsf{S},\mathsf{R})$ *consisting of two stages. Throughout, both parties receive the security parameter* 1^n *as input.*

> Commit. *The sender* S *has a private input* $b \in \{0,1\}$*, which she wishes to commit to the receiver* R*, and a sequence of coin tosses* σ*. At the end of this stage, both parties receive as common output a* commitment z*.*
>
> Reveal. *Both parties receive as input a commitment* z*.* S *also receives the private input* b *and coin tosses* σ *used in the commit stage. After the interaction of* $(\mathsf{S}(b,r),\mathsf{R})(z)$*,* R *either outputs a bit, or the reject symbol* $\perp$*.*

The commitment is public-coin *if the messages the receiver sends are merely the coins it flips at each round.*

[16] We present the definition for bit commitment. To commit to multiple bits, we may simply run a bit commitment scheme in parallel.

For the sake of this tutorial, we focus on commitment schemes with a generic *reveal scheme: the commitment z is simply the transcript of the commit stage, and in the noninteractive reveal stage,* S *sends* (b, σ) *to* R*, and* R *outputs* b *if* S*, on input* b *and randomness* σ*, would have acted as the sender did in* z*; otherwise, it outputs* $\perp$.

Commitment schemes have two security properties. The *hiding* property informally says that, at the end of the commit stage, an adversarial receiver has learned nothing about the message m, except with negligible probability. The *binding* property says that, after the commit stage, an adversarial sender cannot produce valid openings for two distinct messages (i.e., to both 0 and 1), except with negligible probability. Both of these security properties come in two flavors—*statistical*, where we require security even against a computationally unbounded adversary, and *computational*, where we only require security against feasible (e.g., polynomial-time) adversaries.

Statistical security is preferable to computational security, but it is impossible to have commitment schemes that are both statistically hiding and statistically binding. In this tutorial, we focus on statistically hiding (and computationally binding) schemes, which are closely connected with the notion of inaccessible entropy generators.

Definition 23. *A commitment scheme* $\mathsf{Com} = (\mathsf{S}, \mathsf{R})$ *is* statistically hiding *if*

Completeness. *If both parties are honest, then for any bit* $b \in \{0,1\}$ *that* S *gets as private input,* R *accepts and outputs* b *at the end of the reveal stage.*

Statistical Hiding. *For every unbounded strategy* $\widetilde{\mathsf{R}}$*, the distributions* $\mathrm{view}_{\widetilde{\mathsf{R}}}((\mathsf{S}(0), \widetilde{\mathsf{R}})(1^n))$ *and* $\mathrm{view}_{\widetilde{\mathsf{R}}}((\mathsf{S}(1), \widetilde{\mathsf{R}})(1^n))$ *are statistically indistinguishable, where* $\mathrm{view}_{\widetilde{\mathsf{R}}}(e)$ *denotes the collection of all messages exchanged and the coin tosses of* $\widetilde{\mathsf{R}}$ *in* e.

Computational Binding. *A* ppt $\widetilde{\mathsf{S}}$ *succeeds in the following game (breaks the commitment) only with negligible probability in* n*:*

- $\widetilde{\mathsf{S}} = \widetilde{\mathsf{S}}(1^n)$ *interacts with an honest* $\mathsf{R} = \mathsf{R}(1^n)$ *in the commit stage, on security parameter* 1^n*, which yields a commitment* z.
- $\widetilde{\mathsf{S}}$ *outputs two messages* τ_0, τ_1 *such that* $\mathsf{R}(z, \tau_b)$ *outputs* b*, for both* $b \in \{0,1\}$.

Com *is* δ-binding *if no* ppt $\widetilde{\mathsf{S}}$ *wins the above game with probability larger than* $\delta(n) + \mathrm{neg}(n)$.

We now discuss the intriguing connection between statistically hiding commitment and inaccessible entropy generators. Consider a statistically hiding commitment scheme in which the sender commits to a message of length k, and suppose we run the protocol with the message m chosen uniformly at random in $\{0,1\}^k$. Then, by the statistical hiding property, the *real entropy* of the message m after the commit stage is $k - \mathrm{neg}(n)$. On the other hand, the computational binding property says that the *accessible entropy* of m after the commit stage is at most $\mathrm{neg}(n)$. This is only an intuitive connection, since we have not discussed real and accessible entropy for protocols, but only for generators. Such definitions can be found in [11], and for them it can be proven that statistical hiding commitments imply protocols in which the real entropy is much larger than the accessible entropy. Here our goal is to establish the converse, namely that a generator with a gap between its real and accessible entropy implies a statistical hiding commitment scheme. The extension of this fact for protocols can be found in [11].

Theorem 12 (Inaccessible entropy to statistically hiding commitment). *Let* $k = k(n)$*,* $s = s(n)$*, and* $\delta = \delta(n)$ *be polynomial-time computable functions. Let* G *be an efficient* $m = m(n)$*-block generator over* $\{0,1\}^s$*. Assume that* G*'s real Shannon entropy is at least* k*, that its accessible max-entropy is at most* $(1-\delta) \cdot k$*, and that* $k\delta \in \omega(\log n / n)$*. Then for any polynomial-time computable* $g = g(n) \in \omega(\log n)$ *with* $g \geq \mathrm{H}_0(G(U_s))$*, there exists an* $O(m \cdot g / \delta k)$*-round, public-coin, statistically hiding and computationally binding commitment scheme. Furthermore, the construction is black box, and on security parameter* 1^n*, the commitment invokes* G *on inputs of length* s.[17]

[17] Given a, per n, polynomial-size advice, the commitment round complexity can be reduced to $O(m)$. See Remark 7 for details.

Combining the above theorem with Theorem 10 reproves the following fundamental result:

Theorem 13 (One-way functions to statistically hiding commitment). *Assume there exists a nonuniformly one-way function $f\colon \{0,1\}^n \mapsto \{0,1\}^n$, then there exists an $O(n^2/\log n)$-round, public-coin statistically hiding and computationally binding commitment scheme. Furthermore, the construction is black box, and on security parameter 1^n, the commitment invokes f on inputs of length n.*[18]

The heart of the proof of Theorem 12 lies in the following lemma.

Lemma 18. *Let $k = k(n) \geq 3n$ be a polynomial-time computable function, let $m = m(n)$, $s = s(n)$, and let G be an efficient m-block generator over $\{0,1\}^s$. Then there exists a polynomial-time, $O(m)$-round, public-coin, commitment scheme Com with the following properties:*

Hiding: *If each block of G has real min-entropy at least k, then Com is statistically hiding.*
Binding: *If the accessible max-entropy of G is at most $m(k-3n)$, then Com is computationally binding.*

Furthermore, on security parameter 1^n, the protocol invokes G on inputs of length s.

We prove a weak version of Lemma 18 in Section 4.4.1, but we first use it for proving Theorem 12.

Proving Theorem 12.

Proof of Theorem 12. We prove Theorem 12 by manipulating the real and accessible entropy of G using the tools described in Section 4.3, and then applying Lemma 18 on the resulting generator.

Truncated sequential repetition: real entropy equalization. In this step we use G to define a generator $\mathsf{G}^{[v]}$ whose *each block has the same amount* of real entropy— the average of the real entropy of the blocks of G. In relative terms, the entropy gap of $\mathsf{G}^{[v]}$ is essentially that of G. We assume without loss of generality that $m(n)$ is a power of two.[19] We apply truncated sequential repetition (see Definition 20) on G with parameter $w = w(n) = \max\{4, \lceil 16g/\delta k\rceil\} \leq \mathrm{poly}(n)$. Lemma 16, taking $d = g$, yields an efficient $m' = m'(n) = (w-1)\cdot m)$-block generator $\mathsf{G}^{[w]}$ such that the following holds:

- *Each block* of $\mathsf{G}^{[w]}$ has real entropy at least $k = k'(n) = k/m$.
- The accessible max-entropy of $\mathsf{G}^{[w]}$ is at most

$$\begin{aligned} a' = a'(n) &= (w-2)\cdot((1-\delta)\cdot k + \log m + 2g + g \\ &\leq (w-2)\cdot(1-\delta)\cdot k + 4g \\ &\leq (w-2)\cdot(1-\delta/2)\cdot k - (w-2)\cdot k\cdot\delta/2 + 4g \\ &\leq (w-2)\cdot(1-\delta/2)\cdot k - w\cdot k\cdot\delta/4 + 4g \\ &\leq (w-2)\cdot(1-\delta/2)\cdot k \\ &< m'\cdot(1-\delta/2)\cdot k'. \end{aligned}$$

Parallel repetition: converting real entropy to min-entropy and gap amplification. In this step we use $\mathsf{G}^{[w]}$ to define a generator $(\mathsf{G}^{[w]})^{\langle v\rangle}$ whose each block has the same amount of *min-entropy*—about v times the per-block entropy of $\mathsf{G}^{[w]}$. The accessible entropy of $(\mathsf{G}^{[w]})^{\langle v\rangle}$ is also about v times that of $\mathsf{G}^{[w]}$. Let $\ell = \ell(n) \in \Omega(\log n)$ be a polynomial-time computable function that bounds the maximal block length of G. We apply the gap amplification transformation (see Definition 21) on $\mathsf{G}^{[w]}$ with $v = v(n) = \max\{24mn/k\delta, \left\lceil c\cdot(\log n\cdot\ell)/k'\delta)^2\right\rceil\}$, for $c > 0$ to be determined by the analysis. Lemma 17 yields an efficient m'-block generator $(\mathsf{G}^{[w]})^{\langle v\rangle}$ with the following properties:

[18] Applying Theorem 12 with the $O(n/\log n)$-block mentioned in Remark 5 yields an $O(n^2/\log^2 n)$-round commitment. This is the best such commitment scheme we know how to build from one-way functions and it is still far from the $(n/\log n)$ lower bound of [15], which we only know how to achieve via nonuniform protocol (see Remark 7).

[19] Adding $2^{\lceil\log m(n)\rceil} - m(n)$ final blocks of constant value transforms a block generator to one whose block complexity is a power of two, while maintaining the same amount of real and accessible entropy.

- Each block of $(\mathsf{G}^{[w]})^{\langle v\rangle}$ has real *min-entropy* at least $k'' = k''(n) = v \cdot k' - O\left(\log(n) \cdot \ell \cdot \sqrt{v}\right)$.
- The accessible max-entropy of $(\mathsf{G}^{[w]})^{\langle v\rangle}$ is at most $a'' = a''(n) = v \cdot a' + d \cdot m'$, for $d = d(n) = n/8k\delta$.

Hence for large enough n, it holds that

$$\begin{aligned}
m' \cdot k'' - a'' &\geq m' \cdot \left(v \cdot k' - O\left(\log(n) \cdot \ell \cdot \sqrt{v}\right)\right) - (v \cdot a' + d \cdot m') \\
&> m' \cdot \left(v \cdot k' - O\left(\log(n) \cdot \ell \cdot \sqrt{v}\right)\right) - \left(v \cdot (m' \cdot (1 - \delta/2) \cdot k') + d \cdot m'\right) \\
&= v \cdot m' \cdot \left(k'\delta/2 - O(\log(n) \cdot \ell/\sqrt{v}) - d/v\right) \\
&\geq v \cdot m' \cdot \left(k'\delta/2 - O(k'\delta/\sqrt{c}) - d/mn\right) \\
&\geq v \cdot m' \cdot (k'\delta/4 - d/mn) \qquad (.41) \\
&= v \cdot (w-1) \cdot (k\delta/4 - d/n) \\
&= v \cdot (w-1) \cdot k\delta/8 \\
&\geq 3m'n.
\end{aligned}$$

Inequality (.41) holds by taking a large enough value of c in the definition of v.

Namely, the overall real entropy of $(\mathsf{G}^{[w]})^{\langle v\rangle}$ is larger than its accessible max-entropy by at least $3m'n$. Hence, by applying Lemma 18 with $(\mathsf{G}^{[w]})^{\langle v\rangle}$ and $k = k''$, we get the claimed $(m' = m \cdot (w-1) = O(m \cdot g/\delta k))$-round, public-coin, statistically hiding and computationally binding commitment. □

Remark 6 (Comparison with the construction of next-block pseudoentropy generators to pseudorandom generators). It is interesting to see the similarity between the manipulations we apply above on the inaccessible entropy generator G to construct statistically hiding commitment, and those applied in Section 3.4 on the next-block pseudoentropy generator to construct a pseudorandom generator. The manipulations applied in both constructions are essentially the same and achieve similar goals: from real entropy to per-block min-entropy whose overall sum is significantly larger than the accessible entropy in the above, and from next-block pseudoentropy to per-block pseudo-min-entropy whose overall sum is significantly larger than the real entropy in Section 3.4. Combining this fact with the similarity in the initial steps of constructing the above generators from one-way functions (inaccessible entropy generator above and next-block pseudoentropy generator in Section 3.4) yields that the structures of the constructions of statistically hiding commitment schemes and pseudorandom generators from one-way functions are surprisingly similar.

Remark 7 (Constant-round and nonuniform commitments). If the generator's number of blocks is constant, one might skip the first "entropy equalizing" step in the proof of Theorem 12 above, and rather apply parallel repetition directly on G, to get a generator as $\mathsf{G}^{[w]}$ above, but for which we do not know the value of the (possibly different) min-entropies of each block. Since G and thus $\mathsf{G}^{[w]}$ have constant number of blocks, applying a variant of Lemma 18 on $\mathsf{G}^{[w]}$ for polynomially many possible values for the min-entropies (up to some $1/\operatorname{poly}$ additive accuracy level) yields polynomially many commitments that are all binding and at least one of them is hiding. Such commitments can then be combined in a standard way to get a single scheme that is statistically hiding and computationally binding.[20]

The equalization step can also be skipped if the amount of real entropy of each block of the m-block generator G is efficiently computable, yielding an $\Theta(m)$-round commitment scheme (rather than the $O(m \cdot \max\{\log n, g/\delta k\})$-round we know how to achieve without this additional property). This argument also yields an $\Theta(m)$-round, nonuniform (the parties use a nonuniform polynomial-size advice per security parameter) commitment scheme, with no additional assumptions on the generator G. Combining with Theorem 10, the latter yields a $\Theta(n/\log n)$-round nonuniform commitment statistically hiding scheme from any one-way function, matching the lower bound of [15].[21]

[20] [11] used a similar approach to transform a constant-round zero-knowledge proof system for NP that remains secure under parallel composition into a constant-round statistically hiding and computationally binding commitment.

[21] The bound of [15] is stated for uniform commitment schemes, but the same bound for nonuniform commitment schemes readily follows from their proof.

4.4.1 Proving a Weaker Variant of Lemma 18

We prove the following weaker variant of Lemma 18.

Lemma 19 (Weaker variant of Lemma 18). *Let $k = k(n) \geq 3n$ be a polynomial-time computable function, let $m = m(n)$, $s = s(n)$, and let* G *be an efficient m-block generator over $\{0,1\}^s$. Then there exists a polynomial-time, $O(m)$-round, public-coin, commitment scheme* Com *with the following properties:*

Hiding (unchanged): If each block of G *has real min-entropy at least k, then* Com *is statistically hiding.*

Binding: If for every efficient, G-consistent generator $\widetilde{G}$ there exists $i = i(n) \in [m]$ such that

$$\Pr_{\boldsymbol{t} \xleftarrow{R} T_{\widetilde{G}}(1^n)}[\mathrm{AccH}^{i}_{\widetilde{G}}(\boldsymbol{t}) > k - 2n] = \mathrm{neg}(n),$$

then Com *is computationally binding.*

Furthermore, on security parameter 1^n, the protocol invokes G *on inputs of length $s(n)$.*

That is, rather than requiring the *overall* accessible entropy of G to be significantly smaller than its real entropy, we require that, for every efficient, G-consistent generator $\widetilde{G}$, there exists a block in which its accessible entropy is significantly smaller than the real entropy of this block. We do know how to construct such a generator from one-way functions, and moreover, as we show below, such a generator implies an $\Theta(1)$-round statistically hiding commitment, which by [15] cannot be constructed black-boxly from one-way functions. Yet, the proof of Lemma 19 given below does capture some of the main ideas of the proof of Lemma 18. In Section 4.4.2, we give more ideas about the proof of Lemma 18.

To keep notation simple, we take the simplifying assumption that G's input length on security parameter n is n, and assume without loss of generality that all its output blocks are of the same length $\ell = \ell(n)$.[22] We omit n from the notation whenever clear from the context.

On the very high level, to prove Lemma 19 we use a random block of G to mask the committed bit. The guarantee about the real entropy of G yields that the resulting commitment is hiding, where the guarantee about G's accessible entropy, yields that the commitment is weakly (i.e., $\Theta(1/m)$) binding. This commitment is then amplified via parallel repetition, into a full-fledged computationally binding and statistically hiding commitment.

In more detail, the construction of the aforementioned weakly binding commitment scheme goes as follows: The receiver R sends uniformly chosen $i^* \in [m]$ to S. The sender S starts (privately) computing a random execution of G, and sends the first $i-1$ output blocks to R. Then the parties interact in a (constant round) "interactive hashing" subprotocol in which S's input is the i-th block y_i of G. This subprotocol has the following properties:

- After seeing $y_1, \ldots, y_{i-1}$ and the hash value of y_i (i.e., the transcript of the hashing protocol), the (real) min-entropy of y_i is still high (e.g., $\Omega(n)$), and
- If the accessible max-entropy of G in the i-th block is lower than $k - 2n$ (i.e., given an adversarial generator view, the support size of y_i is smaller than 2^{k-2n}), then y_i is *determined* from the point of view of (even a cheating) S after sending the hash value.

Next, S "commits" to its secret bit b by masking it (via XORing) with a bit extracted (via an inner product with a random string) from y_i, and the commit stage halts.

The hiding of the above scheme follows from the guarantee about the min-entropy of G's blocks. The $1/m$-binding of the scheme follows since the bound on the accessible max-entropy of G yields that the accessible entropy of at least one of G's blocks is low, and thus the sender is bounded to a single bit if the receiver has chosen this block to use for the commitment.

The aforementioned hashing protocol is defined and analyzed in Section 4.4.1.1, the weakly binding commitment is defined in Section 4.4.1.2, and in Section 4.4.1.3 we put it all together to prove the lemma.

[22] Using padding technique one can transform a block generator to one whose all blocks are of the same length, without changing its real and its accessible entropy.

4.4.1.1 The Interactive Hashing Protocol

The hashing protocol is the interactive hashing protocol of Ding et al. [6]. (This very protocol is used as the first step of the *computational* interactive hashing protocol used in the commitment constructed in the proof of Lemma 18.)

Let $\mathcal{H}^1 \colon \{0,1\}^\ell \mapsto \{0,1\}^\ell$ and $\mathcal{H}^2 \colon \{0,1\}^\ell \mapsto \{0,1\}^n$ be function families.

Protocol 14 (Two-round interactive hashing protocol $(\mathsf{S_{IH}}, \mathsf{R_{IH}})^{\mathcal{H}_1,\mathcal{H}_2}$)**.**
$\mathsf{S_{IH}}$*'s private input:* $x \in \{0,1\}^\ell$

1. $\mathsf{R_{IH}}$ *sends* $h^1 \xleftarrow{R} \mathcal{H}^1$ *to* $\mathsf{S_{IH}}$.
2. $\mathsf{S_{IH}}$ *sends* $y^1 = h^1(x)$ *back to* $\mathsf{R_{IH}}$.
3. $\mathsf{R_{IH}}$ *sends* $h^2 \xleftarrow{R} \mathcal{H}^2$ *to* $\mathsf{S_{IH}}$.
4. $\mathsf{S_{IH}}$ *sends* $y^2 = h^2(x)$ *back to* $\mathsf{R_{IH}}$.

We will use two properties of the above protocol. The first, which we will use for "hiding", is that $\mathsf{S_{IH}}$ sends only $\ell + n$ bits to $\mathsf{R_{IH}}$. Thus, if $\mathsf{S_{IH}}$'s input x comes from a distribution of min-entropy significantly larger than $\ell + n$, it will still have high min-entropy conditioned on $\mathsf{R_{IH}}$'s view of the protocol (with high probability). On the other hand, the following "binding" property says that, if x has max-entropy smaller than ℓ (i.e., is restricted to come from a set of size at most 2^ℓ) and $\mathcal{H}_1$ and $\mathcal{H}_2$ are "sufficiently" independent, then after the interaction ends, x will be uniquely determined, except with exponentially small probability.

The following proposition readily follows from the proof of [6, Theorem 5.6]:

Proposition 1 ([6], "statistical binding" property of $(\mathsf{S_{IH}}, \mathsf{R_{IH}})$). *Let* $\mathcal{H}^1 \colon \{0,1\}^\ell \mapsto \{0,1\}^\ell$ *and* $\mathcal{H}^2 \colon \{0,1\}^\ell \mapsto \{0,1\}^n$ *be* ℓ*-wise and* 2*-wise independent hash function families, respectively, and let* $\mathcal{L} \subseteq \{0,1\}^\ell$ *be a set of size at most* 2^ℓ. *Let* $\mathsf{S}^*_{\mathsf{IH}}$ *be an (unbounded) adversary playing the role of* $\mathsf{S_{IH}}$ *in* $(\mathsf{S_{IH}}, \mathsf{R_{IH}})$ *that, following the protocol's interaction, outputs two strings* x_0 *and* x_1. *Then, the following holds with respect to a random execution of* $(\mathsf{S_{IH}}, \mathsf{R_{IH}})^{\mathcal{H}^1,\mathcal{H}^2}$*:*

$$\Pr[x_0 \neq x_1 \in \mathcal{L} \ \wedge \ \forall j \in \{0,1\} \colon \ h^1(x_j) = y^1 \wedge h^2(x_j) = y^2] < 2^{-\Omega(n)}.$$

4.4.1.2 Constructing Weakly Binding Commitment

Let $\mathcal{H}^1 = \{\mathcal{H}^1_n = \{h^1 \colon \{0,1\}^{\ell(n)} \mapsto \{0,1\}^{k(n)-2n}\}\}_{n\in\mathbb{N}}$ and $\mathcal{H}^2 = \{h^2_n = \{h^2 \colon \{0,1\}^{\ell(n)} \mapsto \{0,1\}^n\}\}_{n\in\mathbb{N}}$ be function families. Let $\mathsf{G} \colon \{0,1\}^n \mapsto (\{0,1\}^{\ell(n)})^{m(n)}$ be an m-block generator. The weakly binding commitment is defined as follows:

Protocol 15 (Weakly binding commitment scheme $\mathsf{Com} = (\mathsf{S}, \mathsf{R})$)**.**

Common input: *security parameter* 1^n
S*'s private input:* $b \in \{0,1\}$
Commit stage:

1. R *sends* $i^* \xleftarrow{R} [m(n)]$ *to* S.
2. S *starts an execution of* $\mathsf{G}(r)$ *for* $r \xleftarrow{R} \{0,1\}^n$, *and sends* $(y_1, \ldots, y_{i^*-1}) = \mathsf{G}(r)_{1,\ldots,i^*-1}$ *to* R.
3. *The two parties interact in* $(\mathsf{S_{IH}}(y_{i^*} = \mathsf{G}(r)_{i^*}), \mathsf{R_{IH}})^{\mathcal{H}^1_n,\mathcal{H}^2_n}$, *with* S *and* R *taking the roles of* $\mathsf{S_{IH}}$ *and* $\mathsf{R_{IH}}$, *respectively.*
4. S *samples* $u \xleftarrow{R} \{0,1\}^{\ell(n)}$ *and sends* $(\langle u, y_{i^*}\rangle_2 \oplus b, u)$ *to* R, *for* $\langle \cdot,\cdot \rangle_2$ *being inner product modulo* 2.

It is clear that, if $\mathcal{H}_1$, $\mathcal{H}_2$ and G are efficiently computable, then so is Com. We next prove the hiding and binding properties of Com.

Claim 16 (Hiding)**.** *If each block of* G *has real min-entropy at least* k, *then* Com *is statistically hiding.*

Proof. We fix $n \in \mathbb{N}$ and omit it from the notation. For $i \in [m]$, let Y_i denote the i-th block of $\mathsf{G}(U_n)$. By assumption, $\Pr_{\mathbf{y} \overset{\mathrm{R}}{\leftarrow} Y_{1,\ldots,i}}\left[\mathrm{H}_{Y_{i^*}|Y_{<i}}(\mathbf{y}_{i^*}|\mathbf{y}_{<i}) < k\right] = \mathrm{neg}(n)$. It follows (by Lemma 3) that there exists a distribution $(Y_{<i}, Y'_i)$ that is statistically indistinguishable from $(Y_{<i}, Y_i)$, and $Y_i\mid_{Y_{<i}=\mathbf{y}}$ has min-entropy at least k for every value $\mathbf{y} \in \mathrm{Supp}(Y_{<i})$.

Let $\widetilde{\mathsf{R}}$ be an arbitrary algorithm playing the role of R in Com, let $i^* \in [m]$ be its first message and let $V_{i^*}^{\widetilde{\mathsf{R}}}$ be $\widetilde{\mathsf{R}}$'s view in a random execution of $(\mathsf{S}, \widetilde{\mathsf{R}})$, right after S sends the first (i^*-1) output blocks of G. Since $V_{i^*}^{\widetilde{\mathsf{R}}}$ is a probabilistic function of the first (i^*-1) output blocks of G, the distribution of $(V_{i^*}^{\widetilde{\mathsf{R}}}, Y_{i^*})$ is statistically indistinguishable from $(V_{i^*}^{\widetilde{\mathsf{R}}}, Y'_{i^*})$, and $Y'_{i^*}\mid_{V_{i^*}^{\widetilde{\mathsf{R}}}=v_{i^*}^{\widetilde{\mathsf{R}}}}$ has min-entropy at least k for every $v_{i^*}^{\widetilde{\mathsf{R}}} \in \mathrm{Supp}(V_{i^*}^{\widetilde{\mathsf{R}}})$.

Let V be the messages sent by S in embedded execution of the interactive hashing $(\mathsf{S}, \widetilde{\mathsf{R}})$. Since $|V| = k-2n$, it follows (by Lemmas 2 and 3) that $(V_{i^*}^{\widetilde{\mathsf{R}}}, V, Y_{i^*})$ is $(\mathrm{neg}(n) + 2^{-\Omega(n)})$-close to a distribution $(V_{i^*}^{\widetilde{\mathsf{R}}}, V, Y''_{i^*})$, for which $Y''_{i^*}\mid_{(V_{i^*}^{\widetilde{\mathsf{R}}}, H_{i^*})=(v_{i^*}^{\widetilde{\mathsf{R}}}, a_{i^*})}$ has min-entropy at least n for every value $(v_{i^*}^{\widetilde{\mathsf{R}}}, a_{i^*}) \in \mathrm{Supp}(V_{i^*}^{\widetilde{\mathsf{R}}}, V)$. Finally, by the leftover hash lemma (Lemma 8) and the two-universality of the family $\{h_u(y) = \langle u, y\rangle_2 \colon u \in \{0,1\}^n\}$, it holds that $\mathsf{view}^{\widetilde{\mathsf{R}}}(\mathsf{S}(0), \widetilde{\mathsf{R}})$ and $\mathsf{view}^{\widetilde{\mathsf{R}}}(\mathsf{S}(1), \widetilde{\mathsf{R}})$ are of statistical distance at most $\mathrm{neg}(n) + 2^{-\Omega(n)}$, for $\mathsf{view}^{\widetilde{\mathsf{R}}}(\mathsf{S}(b), \widetilde{\mathsf{R}})$ stands for $\widetilde{\mathsf{R}}$'s view at the end, the commit stage interaction $(\mathsf{S}(b), \widetilde{\mathsf{R}})$. □

Claim 17 (Weak binding)**.** *Assume that $\mathcal{H}_1$, $\mathcal{H}_2$ anf* G *are efficiently computable,*[23] *that $\mathcal{H}_1$ and $\mathcal{H}_2$ are ℓ-wise and 2-wise independent, respectively, and that, for every efficient G-consistent generator $\widetilde{G}$, exists $i = i(n) \in [m(n)]$ such that* $\Pr_{\boldsymbol{t} \overset{R}{\leftarrow} T_{\widetilde{G}}(1^n)}\left[\mathrm{AccH}^i_{\widetilde{G}}(\boldsymbol{t}) > k(n) - 2n\right] = \mathrm{neg}(n)$*, then* Com *is* $(1 - 1/3m(n))$*-binding.*

The proof of Claim 17 immediately follows from the next two claims.

Definition 24 (Non-failing senders). *A sender* $\widetilde{\mathsf{S}}$ *is called* non-failing *with respect to a commitment scheme* (S, R)*, if the following holds. Let Z be the transcript of the commit stage of* $(\widetilde{\mathsf{S}}, \mathsf{R})(1^n)$*, and let Σ be the first decommitment string that* $\widetilde{\mathsf{S}}$ *outputs in the (generic) reveal stage, then* $\Pr\left[\mathsf{R}(Z, \Sigma) = \perp\right] = 0$.

That is, a non-failing sender never fails to justify its actions in the commit stage.

Claim 18 (Weak binding against non-failing senders)**.** *Let $\mathcal{H}_1$, $\mathcal{H}_2$ and* G *be as in Claim 17, then* Com *is* $(1 - 1/2m(n))$*-binding against* non-failing *senders.*

Claim 19. *Assume a public-coin commitment scheme is α-binding against non-failing senders, then it is* $(\alpha + \mathrm{neg})$*-binding.*

Proving Claim 18.

Proof. Assume toward a contradiction that there exists a non-failing ppt sender $\widetilde{\mathsf{S}}$ that breaks the $(1 - 1/2m(n))$-binding of Com. We use $\widetilde{\mathsf{S}}$ to construct an efficient adversarial non-failing generator $\widetilde{G}$, such that, for infinitely many n's,

$$\Pr_{(r_1, y_1, w_1, \ldots) \overset{\mathrm{R}}{\leftarrow} T_{\widetilde{G}}(1^n)}\left[\mathrm{H}_{Y_i|R_{<i}}(y_j|r_{<i}) > k(n) - 2n\right] = \Omega(1) \tag{.42}$$

for every $i \in [m(n)]$.

In the following we fix $n \in \mathbb{N}$ on which $\widetilde{\mathsf{S}}$ breaks the binding with probability at least $1 - 1/2m(n)$, and omit n from the notation when clear from the context. We assume for ease of notation that $\widetilde{\mathsf{S}}$ is deterministic. The following generator samples $i \overset{\mathrm{R}}{\leftarrow} [m]$, and then uses the ability of $\widetilde{\mathsf{S}}$ for breaking the binding of the embedded hashing protocol at this round, to output a high-sample-entropy block.

[23] Sampling and evaluation time are polynomial in n.

Algorithm 20 ($\widetilde{G}$—Adversarial cheating generator from cheating sender)**.**
Security paramter: 1^n
Operation:

1. *Let* $i \xleftarrow{R} [m]$.
2. *Emulate a random execution of* $(\widetilde{\mathsf{S}}, \mathsf{R})(1^n)$ *for the first two steps, with* $i^* = i$. *Let* $(y_1, \ldots, y_{i-1})$ *be* $\widetilde{\mathsf{S}}$*'s message in this emulation.*
3. *Output* $y_1, \ldots, y_{i-1}$ *as the first* $i-1$ *output block.*
4. *Continue the emulation of* $(\widetilde{\mathsf{S}}, \mathsf{R})$ *till its end. Let* z *be the transcript of the commit stage, and let* $\sigma_0 = (\cdot, r_0, \cdot)$ *and* $\sigma_1 = (\cdot, r_1, \cdot)$ *be the two strings output by* $\widetilde{\mathsf{S}}$ *at the end of this execution.*
5. *If* $\mathsf{R}(\sigma_1, z) \neq \perp$, *let* $r \xleftarrow{R} \{r_0, r_1\}$. *Otherwise, set* $r = r_0$.
6. *Output* $\mathsf{G}(r)_i, \ldots, \mathsf{G}(r)_m$ *as the last* $m+1-i$ *output block.*

The efficiency of $\widetilde{G}$ is clear, and since $\widetilde{\mathsf{S}}$ is non-failing, it is also clear that $\widetilde{G}$ is G-consistent. In the rest of the proof we show that $\widetilde{G}$ violates the assumed bounds on the (maximal) accessible entropy of $\widetilde{G}$. Specifically, that, for every $i \in [m]$, the sample-entropy of $\widetilde{G}$'s i-th output blocks is larger than $k - 2n$ with probability $\Omega(1/m)$. For ease of notation we prove it for $i = 1$.

Let $\widetilde{T} = (R_1, Y_1, \ldots, R_m, Y_m) = T_{\widetilde{G}}(1^n)$, i.e., a random transcript of $\widetilde{G}$ on security parameter n. Let $\mathcal{Y}$ be the set of all low-entropy first blocks of $\widetilde{G}$. That is,

$$\mathcal{Y} := \{y \colon \mathrm{H}_{Y_1}(y) \leq k - 2n\}.$$

Let Z, $\Sigma_0 = (\cdot, R_0, \cdot)$, and $\Sigma_1 = (\cdot, R_1, \cdot)$, be the value of the strings z, σ_0 and σ_1, respectively, set in Step 4 of $\widetilde{G}$ in the execution described by $\widetilde{T}|_{i=1}$. For $j \in \{0,1\}$, let $Y^j = \mathsf{G}(R_j)_1$ if $\mathsf{R}(Z, \Sigma_j) \neq \perp$, and $\perp$ otherwise. Since $\widetilde{\mathsf{S}}$ (also in the emulated execution done in $\widetilde{G}$) interacts in a random execution of $(\mathsf{S}_{\mathsf{IH}}, \mathsf{R}_{\mathsf{IH}})$, Proposition 1 yields that

$$\Pr\left[\{Y^0, Y^1\} \subseteq \mathcal{Y} \wedge Y^0 \neq Y^1\right] < 2^{-\Omega(n)}. \tag{.43}$$

In addition, since $\widetilde{\mathsf{S}}$ breaks the binding with probability $1 - 1/2m$, it does so with probability at least $1/2$ when conditioning on $i^* = 1$. This yields that

$$\Pr\left[\perp \notin \{Y^0, Y^1\} \wedge Y^0 \neq Y^1\right] \geq 1/2. \tag{.44}$$

We conclude that

$$\begin{aligned}
\Pr\left[Y_1 \notin \mathcal{Y}\right] &\geq \Pr\left[i = 1\right] \cdot 1/2 \cdot \Pr\left[\perp \notin \{Y^0, Y^1\} \wedge \{Y^0, Y^1\} \nsubseteq \mathcal{Y}\right] \\
&\geq 1/2m \cdot \Pr\left[\perp \notin \{Y^0, Y^1\} \wedge \{Y^0, Y^1\} \nsubseteq \mathcal{Y} \wedge Y^0 \neq Y^1\right] \\
&\geq 1/2m \cdot \left(\tfrac{1}{2} - 2^{-\Omega(n)}\right) \geq \Omega(1/m).
\end{aligned}$$

Namely, with probability $\Omega(1/m)$, the accessible entropy of $\widetilde{G}$'s first block is larger than $k - 2n$. Since this holds for any of the blocks, it contradicts the assumption about the accessible entropy of G. □

Proving Claim 19.

Proof. The proof follows a standard argument. Let $\mathsf{Com} = (\mathsf{S}, \mathsf{R})$ be a public-coin commitment scheme, and assume there exists an efficient cheating sender $\widetilde{\mathsf{S}}$ that breaks the binding of Com with probability at least $\alpha(n) + 1/p(n)$, for some $p \in \operatorname{poly}$ and infinitely many n's. We construct an efficient non-failing sender $\widehat{\mathsf{S}}$ that breaks the binding of Com with probability $\alpha(n) + 1/2p(n)$, for infinitely many n's. It follows that if Com is $\alpha(n)$-binding for non-failing senders, then it is $(\alpha(n) + \operatorname{neg}(n))$-binding.

We assume for simplicity that $\widetilde{\mathsf{S}}$ is deterministic, and define the non-failing sender $\widehat{\mathsf{S}}$ as follows: $\widehat{\mathsf{S}}$ starts acting as $\widetilde{\mathsf{S}}$, but before forwarding the i-th message y_i from $\widetilde{\mathsf{S}}$ to R, it first makes sure it will be able to "justify" this message —to output an input for S that is consistent with y_i, and the message $y_1, \ldots, y_{i-i}$ it sent in the previous rounds. To find such a justification string, $\widehat{\mathsf{S}}$ continues, in its head, the interaction between the emulated $\widetilde{\mathsf{S}}$ and R till its end, using fresh coins for the receiver's messages. Since the receiver is public-coin, this efficient random continuation has the same distribution as a (real) random continuation of $(\widetilde{\mathsf{S}}, \mathsf{R})$ has. The sender $\widehat{\mathsf{S}}$ applies such random continuations polynomially many times, and if following one of them $\widetilde{\mathsf{S}}$ outputs a valid decommitment string (which by definition is a valid justification string), it keeps it for future use, and outputs y_i as its i-th message. Otherwise (i.e., it failed to find a justification string for y_i), $\widehat{\mathsf{S}}$ continues as the honest S whose coins and input bit are set to the justification string $\widehat{\mathsf{S}}$ found in the previous round.

Since $\widehat{\mathsf{S}}$ maintains the invariant that it can always justify its messages, it can also do that at the very end of the commitment stage, and thus outputting this string makes it a non-failing sender. In addition, note that $\widehat{\mathsf{S}}$ only fails to find a justification string if $\widetilde{\mathsf{S}}$ has a very low probability to open the commitment at the end of the current interaction, and thus very low probability to cheat. Hence, deviating from $\widetilde{\mathsf{S}}$ on such transcripts will only slightly decrease the cheating probability of $\widehat{\mathsf{S}}$ compared with that of $\widetilde{\mathsf{S}}$.

Assume for concreteness that R sends the first message in Com. The non-failing sender $\widehat{\mathsf{S}}$ is defined as follows:

Algorithm 21 (Non-failing sender $\widehat{\mathsf{S}}$ from failing sender $\widetilde{\mathsf{S}}$)**.**
Input: 1^n
Operation:

```
1. Set $w = (0^{s(n)}, 0)$, for $s(n)$ being a bound on the number of coins used by $\mathsf{S}$, and set $\mathsf{Fail} = \mathsf{false}$.
2. Start an execution of $\widetilde{\mathsf{S}}(1^n)$.
3. Upon getting the $i$-th message $q_i$ from $\mathsf{R}$, do:

    a. If $\mathsf{Fail} = \mathsf{false}$,
        i. Forward $q_i$ to $\widetilde{\mathsf{S}}$, and continue the execution of $\widetilde{\mathsf{S}}$ till it sends its $i$-th message.
        ii. // Try and get a justification string for this $i$-th message.
            Do the following for $3np(n)$ times:
            A. Continue the execution of $(\widetilde{\mathsf{S}}, \mathsf{R})$ till its end, using uniform random messages for $\mathsf{R}$.
            B. Let $z'$ and $w'$ be the transcript and first message output by $\widetilde{\mathsf{S}}$, respectively, at the end of this
               execution.
            C. Rewind $\widetilde{\mathsf{S}}$ to its state right after sending its $i$-th message.
            D. // Update the justification string.
               If $\mathsf{R}(z', w') \neq \perp$. Set $w = w'$ and break the loop.
        iii. If the maximal number of attempts has been reached, set $\mathsf{Fail} = \mathsf{true}$.
    b. // Send the $i$-th message to R. If $\mathsf{Fail} = \mathsf{false}$, this will be the message sent by $\widetilde{\mathsf{S}}$ in Step $3(a)$. Otherwise,
       the string will be computed according to the justification string found in a previous round.
       Send $a_i$ to $\mathsf{R}$, for $a_i$ being the $i$-th message that $\mathsf{S}(1^n, w)$ sends to $\mathsf{R}$ upon getting the first $i$ messages
       sent by $\mathsf{R}$.

4. If $\mathsf{Fail} = \mathsf{false}$, output the same value that $\widetilde{\mathsf{S}}$ does at the end of the execution.
   Otherwise, output $w$.
```

It is clear that $\widehat{\mathsf{S}}$ is non-failing and runs in polynomial time. It is left to argue about its success probability in breaking the binding of Com. We do that by coupling a random execution of $(\widehat{\mathsf{S}}, \mathsf{R})$ with that of $(\widetilde{\mathsf{S}}, \mathsf{R})$, by letting R send the same, uniformly chosen, messages in both executions. We will show that the probability that $\widetilde{\mathsf{S}}$ breaks the binding, but $\widehat{\mathsf{S}}$ fails to do so, is at most $1/3p(n) + m \cdot 2^{-n}$, for m being the round complexity of Com. If follows that, for infinitely many n's, $\widehat{\mathsf{S}}$ breaks the binding of Com with probability $\alpha(n) + 1/2p(n)$.

Let δ_i denote the probability of $\widetilde{\mathsf{S}}$ to break the binding after sending its i-th message, where the probability is over the messages to be sent by R in the next rounds. By definition of $\widehat{\mathsf{S}}$, the probability that $\delta_i \geq 1/3p(n)$ for all $i \in [m]$, and yet $\widehat{\mathsf{S}}$ set $\mathsf{Fail} = \mathsf{true}$, is at most $m \cdot 2^{-n}$. We conclude that the probability that $\widehat{\mathsf{S}}$ does not break the commitment, and yet $\widetilde{\mathsf{S}}$ does, is at most $1/2p(n) + m \cdot 2^{-n}$. □

4.4.1.3 Putting It Together

Given the above, we prove Lemma 19 as follows:

Proof of Lemma 19. Recall that we assume without loss of generality that G's blocks are all of the same length ℓ. We use efficient ℓ-wise function family $\mathcal{H}^1 = \{\mathcal{H}^1_n = \{h^1\colon \{0,1\}^{\ell(n)} \mapsto \{0,1\}^{k(n)-2n}\}\}_{n\in\mathbb{N}}$ and 2-wise function family $\mathcal{H}^2 = \{h^2_n = \{h^2\colon \{0,1\}^{\ell(n)} \mapsto \{0,1\}^n\}\}_{n\in\mathbb{N}}$ (see [4, 30] for constructions of such families).

Claims 16 and 17 yield that the invocation of Protocol 15 with the generator G and the above function families is an $O(m)$-round, public-coin commitment scheme Com that is statistically hiding if the real entropy of G is sufficiently large, and is $(1 - \Theta(1/m))$-binding if the generator max accessible entropy in one of the blocks is sufficiently small. Let $\mathsf{Com}^{\langle m^2\rangle} = (\mathsf{S}^{\langle m^2\rangle}, \mathsf{R}^{\langle m^2\rangle})$ be the m^2 parallel repetition of Com: an execution of $(\mathsf{S}^{\langle m^2\rangle}(b), \mathsf{R}^{\langle m^2\rangle})(1^n)$ consists of $m(n)^2$-fold parallel and independent executions of $(\mathsf{S}(b), \mathsf{R})(1^n)$. It is easy to see that $\mathsf{Com}^{\langle m^2\rangle}$ is statistically hiding since Com is statistically hiding, and by [18] it is computationally binding since Com is $(1 - \Theta(1/m))$-binding. □

4.4.2 About Proving Lemma 18

Recall that the weak binding of Protocol 15 is only guaranteed to hold if the underlying generator G has the following property: any non-failing cheating generator has a round in which its accessible entropy is much smaller than the real entropy G has in this round. However, as we mentioned before, building a generator with this property from one-way functions is beyond the reach of our current techniques, and is impossible to do in a black-box manner. Rather, the type of generators we do know how to build from arbitrary one-way functions are the ones assumed in the statement of the Lemma 18: the *sum* of accessible max-entropy achieved by a cheating non-failing generator is smaller than the *sum* of real entropies (of G). For the latter type of generators, a cheating generator might have high accessible entropy, i.e., as high as the real entropy of G, in *any* of the rounds (though not in all of them simultaneously). In particular, knowing i^*, the sender can put a lot of entropy in i^*'s block. To address this issue, we change the protocol so that the receiver reveals the value of i^* only *after* the interactive hashing protocol. Our hope is that, for at least one value of i, the sender must use a "low-entropy" value y_i in the interactive hashing, and thus we get a binding commitment with probability at least $1/m$. Specifically, consider the following protocol:

Protocol 22 (Commitment scheme $\mathsf{Com} = (\mathsf{S}, \mathsf{R})$, hidden i^*)**.**

Common input: *security parameter* 1^n
S*'s private input:* $b \in \{0,1\}$
Commit stage:

1. R *samples* $i^* \overset{R}{\leftarrow} [m(n)]$.
2. S *starts (internally) an execution of* $\mathsf{G}(r)$ *for* $r \overset{R}{\leftarrow} \{0,1\}^n$.
3. *For* $i = 1$ *to* $m(n)$
   a. *The two parties interact in* $(\mathsf{S}_{\mathsf{IH}}(y_i = \mathsf{G}(r)_i), \mathsf{R}_{\mathsf{IH}})^{\mathcal{H}^1_n, \mathcal{H}^2_n}$*, with* S *and* R *taking the roles of* S_{IH} *and* R_{IH}*, respectively.*
   b. R *informs* S *whether* $i^* = i$.[24]

[24] As defined, R is not public-coin. This, however, is easy to change, without harming the protocol's security, by letting R choose the value of i^* during the execution of the protocol using public coins. I.e., if not set before, at round i it sets i^* to be i with probability $1/(m + 1 - i)$.

c. If informed that $i \neq i^*$, S sends y_i to R.
Otherwise,
i. S samples $u \overset{R}{\leftarrow} \{0,1\}^{\ell(n)}$ and sends $(\langle u, y_i\rangle_2 \oplus b, u)$ to R.
ii. The parties end the execution.

Unfortunately, the basic interactive hashing protocol does not force the sender to decide whether y_i is a low-entropy string or not during the execution of the interactive hashing protocol, and a cheating sender can decide about that *after* it finds out that $i^* = i$. In more detail, given $\mathbf{y}_{<i} = (y_1, \ldots, y_{i-1})$, let $\mathcal{Y}_{\mathbf{y}_{<i}}$ be the set of low-entropy values for y_i conditioned on $\mathbf{y}_{<i}$. (This is defined with respect to a particular cheating sender strategy $\widetilde{\mathsf{S}}$). Since there are not too many high probability distinct strings, the set $\mathcal{Y}_{\mathbf{y}_{<i}}$ is "small". Hence, the interactive hashing guarantees that the probability that the sender can produce *two* distinct elements of $\mathcal{Y}_{\mathbf{y}_{<i}}$ that are consistent with the protocol is negligible. However, it does allow the possibility that the sender can run the interactive hashing protocol consistently with some $y_i \in \mathcal{Y}_{\mathbf{y}_{<i}}$ and afterwards produce a different string y_i that is *not* in $\mathcal{Y}_{\mathbf{y}_{<i}}$, but is consistent with the interactive hashing protocol. This enables a sender to break the binding of the above as follows: consider a cheating sender that runs the generator honestly to obtain $(y_1, \ldots, y_m)$ and uses y_i in the interactive hashing in round i. (Many of these will be low-entropy strings, since the sender is not using any fresh randomness to generate each block.) Upon finding out that the y_i will be used for the commitment, the sender finds another string y'_i (not in $\mathcal{Y}_{\mathbf{y}_{<i}}$) that is consistent with the transcript of the interactive hashing protocol. With these two strings, the sender can now produce a commitment that can be opened in two ways. (Namely, choose u so that $\langle u, y_i\rangle_2 \neq \langle u, y'_i\rangle_2$, and send $(\langle u, y_i\rangle_2, u)$, which also equals $(\langle u, y'_i\rangle_2 \oplus 1, u)$ to the receiver.)

This problem can be solved by using a different interactive hashing protocol that makes it infeasible for the receiver to produce two distinct strings consistent with the protocol where even just *one* of the strings is in a small set $\mathcal{L}$. The new protocol is simply the interactive hashing protocol used above, followed by the sender sending $f(y_i)$ to the receiver, for f being a random member of a *universal one-way hash function* [26] chosen by the receiver. By Rompel [27] (see also [25, 12]) such universal one-way hash functions can be constructed, in a black-box way, from any one-way function, and thus by Lemma 15, they can be constructed from G. The binding of the new interactive hashing protocol is only computational (i.e., unbounded sender can find a collision), compared with the information-theoretic security of the previous interactive hashing protocol, but since the guarantee on the inaccessible entropy of G holds only against computationally bounded entities, this change does not matter to us. The full details of the aforementioned computationally secure interactive hashing protocol and the security proof of the resulting commitment scheme can be found in [16].

Acknowledgment

This tutorial is based on the work we did with Omer Reingold and Hoeteck Wee on inaccessible entropy [16] and with Omer on next-block pseudoentropy [14], and on the understanding developed through the numerous meetings different subsets of the four of us had through the years. We are grateful to Omer and Hoeteck for this collaboration. We also thank Caf Fixe for the excellent coffee that fueled the two of us through much of this research.

We each have individual remarks for Oded.

Iftach: I would like to thank Oded for teaching me the foundation of cryptography course in the most exciting way at the very beginning of my master's studies, and for being my very devoted and inspiring Masters adviser. Without Oded, I would probably not be doing cryptography, and possible not even doing research at all.

Salil: I am indebted to Oded for teaching me many things about research (and life in general). Most relevant to this tutorial, he guided me as I began to study the theory of pseudorandomness, and emphasized the value in seeking simpler and more intuitive proofs and expositions of important results (which motivates much of the work described in this tutorial, not to mention the tutorial itself).

References

[1] B. Barak, R. Shaltiel, and A. Wigderson. Computational analogues of entropy. In *RANDOM-APPROX*, 2003.

[2] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo random bits. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 112–117, 1982.

[3] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.

[4] L. J. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, pages 143–154, 1979.

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

[6] Y. Z. Ding, D. Harnik, A. Rosen, and R. Shaltiel. Constant-round oblivious transfer in the bounded storage model. In *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004*, pages 446–472, 2004.

[7] 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.

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

[9] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, pages 270–299, 1984.

[10] 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.

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

[12] I. Haitner, T. Holenstein, O. Reingold, S. Vadhan, and H. Wee. Universal one-way hash functions via inaccessible entropy. In *Advances in Cryptology – EUROCRYPT 2010*, pages 616–637, 2010.

[13] I. Haitner, D. Harnik, and O. Reingold. On the power of the randomized iterate. *SIAM J. Comput.*, 40 (6):1486–1528, 2011. Preliminary version in *Crypto'06*.

[14] I. Haitner, O. Reingold, and S. Vadhan. Efficiency improvements in constructing pseudorandom generators from one-way functions. *SIAM Journal on Computing*, 42(3):1405–1430, 2013.

[15] I. Haitner, J. J. Hoch, O. Reingold, and G. Segev. Finding collisions in interactive protocols—tight lower bounds on the round and communication complexities of statistically hiding commitments. *SIAM Journal on Computing*, 44(1):193–242, 2015.

[16] I. Haitner, O. Reingold, S. Vadhan, and H. Wee. Inaccessible entropy i: I.e. generators and statistically hiding commitments from one-way functions. www.cs.tau.ac.il/~iftachh/papers/AccessibleEntropy/IE1.pdf, 2016. To apper. Prelimanry version,named Inaccessible Entropy, aperared in STOC 09.

[17] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, pages 1364–1396, 1999.

[18] J. Hästad, R. Pass, K. Pietrzak, and D. Wikström. An efficient parallel repetition theorem. In *Theory of Cryptography, Sixth Theory of Cryptography Conference, TCC 2010*, 2010.

[19] 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.

[20] T. Holenstein and M. Sinha. Constructing a pseudorandom generator requires an almost linear number of calls. In *Proceedings of the 53rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 698–707, 2012.

[21] C.-Y. Hsiao, C.-J. Lu, and L. Reyzin. Conditional computational entropy, or toward separating pseudoentropy from compressibility. In *Advances in Cryptology – EUROCRYPT 2007*, pages 169–186, 2007.

[22] R. Impagliazzo and M. Luby. One-way functions are essential for complexity based cryptography. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 230–235, 1989.

[23] 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.

[24] 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.

[25] J. Katz and C. Koo. On constructing universal one-way hash functions from arbitrary one-way functions. Technical Report 2005/328, Cryptology ePrint Archive, 2005.

[26] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 33–43, 1989.

[27] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 387–394, 1990.

[28] C. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, pages 656–715, 1949.

[29] S. Vadhan and C. J. Zheng. Characterizing pseudoentropy and simplifying pseudorandom generator constructions. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, pages 817–836, 2012.

[30] M. N. Wegman and J. L. Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 1981.

[31] G. Yang. *Cryptography and Randomness Extraction in the Multi-Stream Model*. PhD thesis, Tsinghua University, Beijing, China, 2015. http://eccc.hpi-web.de/static/books/Cryptography_and_Randomness_Extraction_in_the_Multi_Stream_Model.

[32] A. C. Yao. Theory and applications of trapdoor functions. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 80–91, 1982.

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