25 Computational Entropy

Salil Vadhan

25.1 Introduction

The foundations of cryptography laid by Shafi Goldwasser and Silvio Micali in the 1980s provided remarkably strong security definitions (e.g., semantic security Goldwasser and Micali [1984]) and amazingly rich cryptographic functionalities (e.g., zero-knowledge proofs Goldwasser et al. [1988]) that could be achieved from precisely stated complexity assumptions (e.g., the quadratic residuosity assumption [Goldwasser and Micali 1984]). This naturally led to an important project of understanding what are the *minimal* complexity assumptions needed to each of the many cryptographic primitives that were emerging.

The pinnacle of success in this effort was to show that a given cryptographic primitive could be based on the existence of mere one-way functions, as defined in the work of Diffie and Hellman [1976] that initiated complexity-based cryptography. The notion of a one-way function is both very general, with many concrete candidates for instantiation, and very simple to state, allowing the candidates to cryptanalyzed more easily. Moreover, almost all primitives in complexity-based cryptography imply the existence of one-way functions [Impagliazzo and Luby 1989], so one-way functions are in some sense the minimal assumption we could hope for.

Remarkably, it was discovered that a wide array of cryptographic primitives could be constructed assuming only the existence one-way functions. These included such powerful objects as chosen-ciphertext-secure symmetric encryption, pseudorandom functions, digital signatures, and zero-knowledge proofs and statistical zero-knowledge arguments for all of **NP** [Goldreich et al. 1986, Goldreich et al. 1991, Håstad et al. 1999, Naor and Yung 1989, Rompel 1990, Naor 1991, Haitner et al. 2009a]. All of these constructions begin by converting the "raw hardness" of a one-way function to one of the following more structured cryptographic primitives: a pseudorandom generator [Blum and Micali 1984, Yao 1982], a universal one-way hash function [Naor and Yung 1989], or a statistically hiding commitment scheme [Brassard et al. 1988].

The goal of this survey is to convey how this conversion from one-wayness to structured hardness is possible, focusing on the cases of constructing pseudorandom generators and statistically hiding commitments. The common answer that has emerged through a series of works is as follows:

1. The security properties of these (and other) cryptographic primitives can be understood in terms of various *computational analogues of entropy*, and in particular how these computational measures of entropy can be very different from real information-theoretic entropy.
2. It can be shown that every one-way function directly exhibits some gaps between real entropy and the various computational entropies.
3. Thus we can construct the desired cryptographic primitives by amplifying and manipulating the entropy gaps in a one-way function, through forms of repetition and hashing.

This viewpoint (of identifying and manipulating computational entropy) was already present in the original constructions of pseudorandom generators, universal one-way hash functions, and statistically hiding commitments from arbitrary one-way functions [Håstad et al. 1999, Rompel 1990, Haitner et al. 2009a], but those constructions were quite complicated and inefficient, making it hard to distinguish the essential ideas from technicalities. Over the past decade, a clearer picture has emerged through the introduction of new, refined notions of computational entropy [Haitner et al. 2009b, Haitner et al. 2013, Haitner et al. 2010, Vadhan and Zheng 2012, Agrawal et al. 2019]. The resulting constructions of pseudorandom generators and statistically hiding commitments from one-way functions are much simpler and more efficient than the original ones, and are based entirely on natural manipulations of computational entropy. The two constructions are "dual" to each other, whereby the construction of pseudorandom generators relies on a form of computational entropy ("pseudoentropy") being larger than the real entropy, while the construction of statistically hiding commitments relies on a form of computational entropy ("accessible entropy") being smaller than the real entropy. Beyond that difference, the two constructions share a common structure, using a very similar sequence of manipulations of real and computational entropy.

In this survey, we will describe the main ideas behind these recent constructions of pseudorandom generators and statistically hiding commitments from one-way functions. We will warm up by "deconstructing" the classic construction of pseudorandom generators from one-way *permutations* [Blum and Micali 1984, Yao 1982, Goldreich and Levin 1989] using the modern language of computational entropy, as it will provide intuition and context for what follows. We will then present the state-of-art construction of pseudorandom generators from general one-way functions, using the computational entropy notions of “conditional KL-hardness” and “next-block pseudoentropy” [Haitner et al. 2013, Vadhan and Zheng 2012]. Finally, we will introduce the dual notion of “next-block accessible entropy” and explain how it is used in constructing statistically hiding commitments from one-way functions in a way that parallels the aforementioned construction of pseudorandom generators [Haitner et al. 2009b].

Beyond the specific constructions covered, we hope that the surveyed notions of computational entropy and the tools for reasoning about them will prove useful elsewhere, for example in some of the other application areas for computational entropy mentioned below.

Other Reading. For general background on the foundations of cryptography and the theory of pseudorandomness, we recommend Goldreich [2019], Goldreich [2008, Ch. 8], and Vadhan [2012]. A more detailed and technical tutorial on the constructions of pseudorandom generators and statistically hiding commitments from one-way functions using computational entropy is given by Haitner and Vadhan [2017]. While we focus on its role in constructions of cryptographic primitives from one-way functions, computational analogues of entropy have been studied from a number of other angles. Yao [1982] introduced a notion of computational entropy for the purposes of studying efficient data compression and error correction. Barak et al. [2003] carry out a systematic study of several different notions of computational entropy (some of which appear here). Forms of computational entropy have also found applications in leakage-resilient cryptography [Dziembowski and Pietrzak 2008], deterministic encryption [Fuller et al. 2015], memory delegation [Chung et al. 2011], and differential privacy [Mironov et al. 2009], and these areas of research have developed the theory of computational entropy in other ways. Recently, Haitner et al. [2018] have introduced a computational analogue of independence for outputs from a 2-party protocol that they use to characterize the 2-party cryptographic primitives whose existence is equivalent to the existence of key agreement protocols.

Acknowledgments. I vividly recall the thrill of learning about the foundations of cryptography from Shafi and Silvio with my fellow graduate students at MIT in the late 1990s. Time and time again, it felt like we were seeing how the seemingly

impossible could be achieved. We saw how philosophical and psychological concepts (e.g., knowledge, persuasion, impersonation) could be given convincing mathematical definitions, and cryptographic schemes for controlling these concepts could be constructed based on simple complexity assumptions. Decades later, the desire to better understand how this all could be possible has remained with me, and is a major motivator for the line of research described in this survey. At a more concrete level, much of this work was a direct outgrowth of the line of research that Shafi started me on as her Ph.D. student—namely, the complexity of statistical zero-knowledge proofs [Vadhan 1999]. Attempting to understand the complexity of the prover in statistical zero-knowledge proofs led to a characterization of statistical zero knowledge in terms of "instance-dependent" commitment schemes [Bellare et al. 1990, Itoh et al. 1997, Micciancio and Vadhan 2003, Nguyen and Vadhan 2006, Ong and Vadhan 2007]. The ideas underlying that characterization inspired the construction of statistically hiding commitments from one-way functions [Haitner et al. 2009a], including the use of computational entropies in that work and the subsequent ones discussed in this survey. Thank you, Shafi and Silvio, for creating and leading us to such a beautiful landscape to explore, and for your mentorship and friendship throughout our lives!

I am grateful to Oded Goldreich for his unwavering support and patience for my writing of this survey, and lots of helpful feedback. Thanks also to Hugo Krawczyk, whose encouragement motivated me to write this survey based on my lectures at the 2016 IACR-COST School on Randomness in Cryptography. The writing of this survey was supported by NSF grant CCF-1763299.

25.2 Basic Information-Theoretic Notions

We review a number of information-theoretic notions, before introducing their computational analogues, which will be the main focus of this survey.

Basic Definitions. We begin with the most intuitive measure of distance between probability distributions:

Definition 25.1 (Statistical difference) Let X and Y be discrete random variables taking values in a universe $\mathcal{U}$. Then the ***statistical difference*** (a.k.a. ***total variation distance***) between X and Y is

$$d(X, Y) = \max_{T \subseteq \mathcal{U}} |\Pr[X \in T] - \Pr[Y \in T]| \in [0, 1].$$

We say X and Y are ε-***close*** if $d(X, Y) \leq \varepsilon$.

We will also discuss a number of different measures of entropy:

Definition 25.2 (Entropy measures) Let X be a discrete random variable. Then:

- The (*Shannon*) *entropy of* X is

$$\mathrm{H}(X) = \mathrm{E}_{x \xleftarrow{R} X}\left[\log\left(\frac{1}{\Pr[X=x]}\right)\right].$$

- The *min-entropy of* X is

$$\mathrm{H}_\infty(X) = \min_x\left[\log\left(\frac{1}{\Pr[X=x]}\right)\right] = \log\left(\frac{1}{\max_x \Pr[X=x]}\right).$$

- The *max-entropy of* X is

$$\mathrm{H}_0(X) = \log|\operatorname{Supp}(X)|.$$

Above, and throughout this survey, all logarithms are base 2 (except where explicitly noted otherwise) and $\operatorname{Supp}(X) = \{x : \Pr[X = x] > 0\}$ denotes the support of the random variable X.

$\mathrm{H}(X)$ measures the *average* number of bits of randomness in X, while $\mathrm{H}_\infty(X)$ and $\mathrm{H}_0(X)$ are worst-case lower and upper bounds on $\mathrm{H}(X)$. Indeed, we have

$$\mathrm{H}_\infty(X) \le \mathrm{H}(X) \le \mathrm{H}_0(X),$$

with equality if and only if X is uniform on $\operatorname{Supp}(X)$; that is, X is a ***flat distribution***.

Extraction and Compression. The usefulness of min-entropy in theoretical computer science was advocated by Chor and Goldreich [1988]. Specifically, having a random variable with high min-entropy is preferable to having a random variable with high Shannon entropy because high min-entropy can be converted into nearly uniform randomness via extractors:

Definition 25.3 (Randomness extractors Nisan and Zuckerman [1996]) A function $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a *strong* (k,ε)*-extractor* if for every random variable X distributed on $\{0,1\}^n$ with $\mathrm{H}_\infty(X) \ge k$, the random variable $(U_d, \mathrm{Ext}(X, U_d))$ is ε-close to (U_d, U_m) where U_d and U_m are uniformly distributed on $\{0,1\}^d$ and $\{0,1\}^m$, respectively, and X, U_d, U_m are mutually independent.

Above, and throughout, when the same random variable appears twice in an expression (e.g., the U_d in $(U_d, \mathrm{Ext}(X, U_d))$), they take the same value with probability 1.

Lemma 25.1 (Leftover hash lemma Bennett et al. [1988], Håstad et al. [1999]) For every n, $k \le n$, and $\varepsilon > 2^{-k/2}$, there is a polynomial-time computable strong (k,ε)-extractor

$\text{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ that has output length $m = \lfloor k - 2\log(1/\varepsilon) \rfloor$ and seed length $d = n$. Specifically, we can take $\text{Ext}(x, h) = h(x)$ where h comes from a 2-universal family of hash functions mapping $\{0,1\}^n$ to $\{0,1\}^m$.

Note that the extractors given by the Leftover Hash Lemma extract almost all of the min-entropy out of the source, except for a $2\log(1/\varepsilon)$ entropy loss, which is necessary for any extractor [Radhakrishnan and Ta-Shma 2000]. The seed length $d = n$, however, is suboptimal, and there is a long line of research on randomness extractors that gives explicit constructions of extractors with seed length depending only logarithmically on n. (See Vadhan [2012, Chapter 6] and the references therein.)

Similarly, having a random variable with low max-entropy is often preferable to having one with low Shannon entropy because low max-entropy allows for "compression".

Lemma 25.2 For every n, $k \le n$, and $\varepsilon > 0$, there is a polynomial-time computable encoding function $\text{Enc} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with output length $m = k + \log(1/\varepsilon)$ and seed length $d = n$ such that for every random variable X distributed on $\{0,1\}^n$ with $\text{H}_0(X) \le k$, there is a (not necessarily efficient) decoding function $\text{Dec} : \{0,1\}^m \times \{0,1\}^d \to \{0,1\}^n$ such that:

$$\Pr\left[\text{Dec}(\text{Enc}(X, U_d), U_d) = X\right] \ge 1 - \varepsilon.$$

Again we can take $\text{Enc}(x, h) = h(x)$ where h comes from a 2-universal family of hash functions mapping $\{0,1\}^n$ to $\{0,1\}^m$.

That is, if the max-entropy is low, then we do not need to reveal much information (just the m bits output by Enc to uniquely determine) to determines x with high probability.

Min-entropy and max-entropy are rather brittle, in that making a small change to the probability distribution can dramatically change the amount of measured entropy. For this reason, it is common to work with "smoothed" forms of these entropies [Renner and Wolf 2005]. Specifically, we consider a random variable X to have ***smoothed min-entropy at least*** k if X is ε-close to a random variable X' with $\text{H}_\infty(X') \ge k$, for a negligible ε. And we consider a random variable X to have ***smoothed max-entropy at most*** k if X is ε-close to a random variable X' with $\text{H}_0(X') \le k$. Notice that smoothed min-entropy and smoothed max-entropy support randomness extraction and compression, as above, with the smoothing error adding to the error parameter of the randomness extraction or decoding.

Conditional Entropies. We will also make use of conditional forms of entropy. For Shannon entropy, there is a standard definition:

Definition 25.4 (Conditional entropy) For jointly distributed discrete random variables (X, Y), the ***conditional Shannon entropy*** of X given Y is

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

where $X|_E$ is the notation we use for conditioning the random variable X on event E.

There are a number of natural ways to define conditional min-entropy and conditional max-entropy, but for the case of min-entropy the following has proved to be particularly convenient in cryptographic applications.

Definition 25.5 (Average min-entropy Dodis et al. [2008]) For jointly distributed discrete random variables (X, Y), the ***average min-entropy*** of X given Y is

$$\mathrm{H}_\infty(X|Y) = \log\left(\frac{1}{\mathrm{E}_{y \overset{R}{\leftarrow} Y}\left[2^{-\mathrm{H}_\infty\left(X|_{Y=y}\right)}\right]}\right).$$

Despite the somewhat complicated definition, average min-entropy has a very natural operational interpretation as measuring the maximum probability of guessing X from Y:

Lemma 25.3 (Guessing min-entropy) For every pair of jointly distributed discrete random variables (X, Y), the average min-entropy $\mathrm{H}_\infty(X|Y)$ equals the ***guessing min-entropy*** of X given Y, defined as

$$\mathrm{H}_{guess}(X|Y) = \log\left(\frac{1}{\max_A \Pr[A(Y) = X]}\right),$$

where the maximum is over all functions A (regardless of computational complexity).

The proof of this lemma follows from observing that $2^{-\mathrm{H}_\infty\left(X|_{Y=y}\right)} = \max_x \Pr[X = x|Y = y]$, which is exactly the success probability of an optimal strategy for guessing X given that $Y = y$.

In addition to having this nice operational interpretation, average min-entropy also supports randomness extraction. Indeed, it turns out that every randomness extractor for ordinary min-entropy is also one for average min-entropy with only a small loss in the error parameter:

Lemma 25.4 (Vadhan [2012, Problem 6.8]) Let $\mathrm{Ext} : \{0, 1\}^n \times \{0, 1\}^d \to \{0, 1\}^m$ be a (k, ε)-extractor for $k \le n - 1$, and let (X, Y) be any pair of jointly distributed discrete random variables with X taking values in $\{0, 1\}^n$ such that $\mathrm{H}_\infty(X|Y) \geq k$. Then $(U_d, \mathrm{Ext}(X, U_d), Y)$ is 3ε-close to (U_d, U_m, Y), where U_d, U_m, and (X, Y) are mutually independent.

The above lemma is proven by showing that on one hand, a (k, ε)-extractor also extracts nearly uniform bits when applied to sources of min-entropy k' slightly smaller than k, and on the other hand, if X has average min-entropy at least k given Y, then $X_{Y=y}$ is very unlikely (over the choice of $y \overset{R}{\leftarrow} Y$) to have min-entropy much smaller than k, In fact, the extractor of the leftover hash lemma can directly be shown to be an extractor for average min-entropy with no loss in the error parameter [Dodis et al. 2008].

Flattening. Although min-entropy and max-entropy are more directly useful in cryptographic applications, many of the results we will discuss will begin by establishing statements involving Shannon entropy. These can converted into statements about (smoothed) min-entropy and max-entropy by taking many independent samples:

Lemma 25.5 (Flattening) Let X be a random variable distributed on $\{0, 1\}^n$, and let X^t consist of t independent samples of X. Then for every $\varepsilon \in (0, 1/2)$, the random variable X^t is ε-close to a random variable X' such that

$$\mathrm{H}_\infty(X') \geq t \cdot \mathrm{H}(X) - O\left(\sqrt{t \cdot \log(1/\varepsilon)} \cdot n\right) \quad \text{and}$$

$$\mathrm{H}_0(X') \leq t \cdot \mathrm{H}(X) + O\left(\sqrt{t \cdot \log(1/\varepsilon)} \cdot n\right).$$

The flattening lemma can be viewed as a quantitative form of the standard "asymptotic equipartition property" in information theory. Various forms of it appear in the literature, including in Håstad et al. [1999]; the tight version above is from Holenstein and Renner [2011].

Note that the Shannon entropy of X^t is exactly $t \cdot \mathrm{H}(X)$, which grows linearly with t. The above lemma says that, after some smoothing, the min-entropy and max-entropy of X^t are close to the Shannon entropy of X^t, with a difference that grows only like $\sqrt{t}$. In particular, for $\varepsilon = n^{-\log n}$ and $t = n^2 \cdot \log^3 n$, the smoothed min- and max-entropies are guaranteed to be $t \cdot (\mathrm{H}(X) \pm o(1))$. This is referred to as a "flattening" lemma because the only random variables where the Shannon entropy equals the min-entropy or max-entropy are flat random variables (ones that are uniform on their support), whereas X^t is close to a distribution in which the min- and max-entropies are relatively close (i.e., are $o(t)$ away). Flattening also works for jointly distributed random variables (X, Y); see Holenstein and Renner [2011] for a precise statement.

25.3 Basic Computational Notions

We review the standard notions of one-way functions, computational indistinguishability, and pseudorandom generators, highlighting notational choices and conventions we will use throughout this survey.

One-Way Functions. A one-way function is a function that is easy to compute in the forward direction, but very hard to invert, even on average.

Definition 25.6 (One-way functions Diffie and Hellman [1976]) A function $f : \{0,1\}^n \to \{0,1\}^n$ is a ***one-way function (OWF)*** if:

1. f is computable in time $\mathrm{poly}(n)$.
2. For some $s(n) = n^{\omega(1)}$ and $\varepsilon(n) = 1/n^{\omega(1)}$, and all nonuniform algorithms A running in time $s(n)$, we have

$$\Pr\left[A(f(X)) \in f^{-1}(f(X))\right] \le \varepsilon(n),$$

where the probability is taken over $X \overset{R}{\leftarrow} \{0,1\}^n$ and the coin tosses of A.

Note that the asymptotics are somewhat hidden in the above definition. As usual, the definition actually refers to an infinite family of functions $\{f_n : \{0,1\}^n \to \{0,1\}^n\}_{n\in\mathbb{N}}$, one for each value of the security parameter n. Condition 1 means that there should be a single uniform algorithm that can evaluate $f = f_n$ in time $\mathrm{poly}(n)$ for all n. On the other hand, we require security to hold even against nonuniform algorithms. We adopt this nonuniform model of security because it simplifies a number of the definitions and proofs, but all of the results we will discuss have uniform-security analogues. The time bound $s(n)$ on the complexity of the nonuniform algorithm A should be interpreted as a bound on both the running time and program size; this is equivalent (up to a polynomial loss) to taking $s(n)$ to be a bound on the size of A as a Boolean circuit.

The security bound $n^{\omega(1)}$ refers to any functions that is asymptotically larger than every polynomial function. It is more common in the literature to state security definitions for cryptographic primitives in the form "for every constant c and every nonuniform algorithm A running in time n^c, the success probability of A in inverting f is at most $1/n^c$ for all sufficiently large n." Definition 25.6 can be shown to be equivalent to such formulations [Bellare 2002]. Note that the functions $s(n)$ and $\varepsilon(n)$ in Definition 25.6 are not necessarily efficiently computable. However, we will ignore that subtlety in this survey, and pretend that they are efficiently computable when it makes the exposition simpler.

Note that we have taken the security parameter n to equal the input and output lengths of the one-way function. When we define other primitives (such as pseudorandom generators below), we will allow their input and output lengths to be polynomially related to the security parameter, rather than equal to the security parameter. This will allow us to have a more fine-grained discussion of the complexity of constructing these primitives from one-way functions, where we will keep the security parameter n equal to the input length of the underlying one-way function.

We stress that Definition 25.6 does not require the function f to be one-to-one, and thus the adversary A succeeds if it finds *any* preimage of its input $f(X)$. Overcoming the challenges introduced by general, many-to-one one-way functions f is a major theme in this survey.

Computational Indistinguishability. The fundamental concept of computational indistinguishability was introduced in the seminal paper of Goldwasser and Micali [1984]. It is the computational analogue of statistical difference (Definition 25.1), obtained by restricting to statistical tests T that are efficiently computable:

Definition 25.7 (Computational indistinguishability Goldwasser and Micali [1984]) Let X and Y be random variables distributed over $\{0,1\}^m$ for $m = \text{poly}(n)$, where n is a security parameter. We say that X and Y are ***computationally indistinguishable,*** written $X \overset{c}{\equiv} Y$, if for some $s(n) = n^{\omega(1)}$ and $\varepsilon(n) = 1/n^{-\omega(1)}$, and all nonuniform algorithms T running in time $s(n)$, we have

$$|\Pr[T(X) = 1] - \Pr[T(Y) = 1]| \leq \varepsilon(n), \qquad (25.1)$$

where the probability is taken over X, Y, and the coin tosses of T. If Y is identically distributed to U_m, the uniform distribution on $\{0,1\}^m$, then we say that X is *pseudorandom.*

If Inequality (25.1) holds for all (computationally unbounded) functions T (i.e., X and Y are $\varepsilon(n)$-close in statistical difference for some $\varepsilon(n) = n^{-\omega(1)}$), then we say that X and Y are *statistically indistinguishable* and write $X \overset{s}{\equiv} Y$.

Computational indistinguishability is the basis of many concepts in modern cryptography, including the fundamental notion of a pseudorandom generator:

Definition 25.8 (Pseudorandom generators Blum and Micali [1984], Yao [1982]) A function $G : \{0,1\}^\ell \to \{0,1\}^m$, where $\ell, m = \text{poly}(n)$ for a security parameter n, is a ***pseudorandom generator (PRG)*** if:

1. G is computable in deterministic time poly(n).
2. $G(U_\ell) \overset{c}{\equiv} U_m$.
3. $m > \ell$.

We call ℓ the *seed length* of G and m the *output length*.

Note that the above definition only requires that the output length is larger than the seed length by at least one bit ($m > \ell$). Many applications of pseudorandom generators require generating many pseudorandom bits from a short seed ($m \gg \ell$). Fortunately, there is a generic length-expansion technique that converts pseudorandom generators that stretch by one bit into ones that stretch by any desired length (without increasing the seed length) [Goldreich and Micali 1984]. Thus in this survey we will not specify the stretch of the pseudorandom generators.

Pseudorandom Generators from One-Way Functions. A celebrated result in the foundations of cryptography is that pseudorandom generators can be constructed from any one-way function.

Theorem 25.1 (PRGs from OWFs Håstad et al. [1999]) If there exists a one-way function $f : \{0, 1\}^n \to \{0, 1\}^n$, then there exists a pseudorandom generator $G^f : \{0, 1\}^\ell \to \{0, 1\}^m$.

The original construction of Håstad et al. [1999] proving Theorem 25.1 was quite complex and inefficient. The pseudorandom generator G^f has a seed length of $\ell = \Theta(n^{10})$ and requires evaluating the one-way function at least $q = \Omega(n^{10})$ times. Quantifying these complexity parameters makes sense because the pseudorandom generator construction is a "(fully) black-box" one [Reingold et al. 2004], where the given one-way function is used as an oracle in the algorithm for computing G (so q counts the number of oracle queries), and the security of the construction is proven via a reduction that uses any oracle T that distinguishes $G^f(U_\ell)$ from U_m to invert f with nonnegligible probability. (The reduction also uses an oracle for f and may use nonuniform advice when working in the nonuniform security model, as we are.)

One might think that the large polynomial complexity of the construction does not matter because the "polynomial security" formulations of Definitions 25.6 and 25.8 are invariant to polynomial changes in the security parameter n. For example, G could invoke f on inputs of length $n^{1/10}$ and thereby achieve seed length $\ell = O(n)$. But this does not really change anything. In either case, the problem is that the security of the pseudorandom generator on seed length ℓ is related to the security of the one-way function on inputs of length $\Theta(\ell^{1/10})$, which amounts to an unsatisfactory loss in security. This becomes even more apparent when quantifying the security more finely. For example, even if the one-way function had "optimal" hardness, with security against algorithms running time $s(n) = 2^{cn}$ for a constant $c > 0$ on inputs of length n, we would only be guaranteed that the pseudorandom generator is secure against algorithms running in time $s(n)^{\Theta(1)} = 2^{\Theta(\ell^{1/9})}$, which is very far from the $2^{c'\ell}$ security that we might hope for. Thus it is important to seek more efficient constructions.

25.4 Pseudoentropy

A pseudorandom generator $G : \{0, 1\}^\ell \to \{0, 1\}^m$ with large stretch ($\ell \ll m$) starkly demonstrates the difference between computational and information-theoretic notions. On one hand, the output distribution $G(U_\ell)$ has entropy at most ℓ (since applying a deterministic function cannot increase entropy), but it is computationally indistinguishable from the distribution on $\{0, 1\}^m$ with maximal entropy—namely, U_m. Thus, as an intermediate step toward constructing pseudorandom generators, it is natural to consider a more quantitative measure of the amount of "computational entropy," as done by Håstad et al. [1999] in their proof of Theorem 25.1:

Definition 25.9 (Pseudoentropy Håstad et al. [1999]) Let X be a random variable distributed on strings of length poly(n) for a security parameter n. We say that X has *pseudoentropy at least k* if there exists a random variable X' such that

1. $X' \overset{c}{\equiv} X$.
2. $\mathrm{H}(X') \geq k$.

If Condition 2 is replaced with $\mathrm{H}_\infty(X') \geq k$, then we say that X has *pseudo-min-entropy at least k*.

As discussed above, constructing a pseudorandom generator requires producing an efficiently samplable distribution whose pseudoentropy (and pseudo-min-entropy) is *larger* than its actual entropy. We have formulated the definition of pseudoentropy to only allow for expressing such lower bounds on computational entropy (e.g., "X has pseudoentropy at least k"). Using the same template as a definition of "pseudoentropy at most k" yields a nonuseful definition, since every random variable can be shown to have pseudoentropy at most polylog(n) via a probabilistic argument akin to the one used in Goldreich and Krawczyk [1992]. In Section 25.7, we shall see a different approach that leads to a useful definition of upper bounds on computational entropy.

The Håstad et al. [1999] notion of pseudoentropy is very useful, thanks to the power of computational indistinguishability, which says that two random variables are essentially equivalent for the purposes of any efficient computation on them. In particular, pseudo-min-entropy supports randomness extraction, by any efficiently computable extractor:

Lemma 25.6 Let X be a random variable distributed on strings of length $m = \text{poly}(n)$ for a security parameter n with pseudo-min-entropy at least k, and let $\text{Ext} : \{0, 1\}^m \times \{0, 1\}^d \to \{0, 1\}^{m'}$ be a strong (k, ε)-extractor computable in time $\text{poly}(n)$, with error $\varepsilon = n^{-\omega(1)}$. Then $(U_d, \text{Ext}(X, U_d))$ is pseudorandom.

In particular, using the leftover hash lemma (Lemma 25.1), the pseudoentropy loss $k - m'$ incurred by extraction is only $2 \log(1/\varepsilon)$, which we can take to be any function that is $\omega(\log n)$.

As with the information-theoretic notions, randomness extraction requires pseudo-*min*-entropy rather than plain pseudoentropy. Fortunately, flattening also works in the context of pseudoentropy, and using Lemma 25.5, it can be shown that if X has pseudoentropy at least k, then for any $t = \text{poly}(n)$, the product X^t has pseudo-min-entropy at least $t \cdot k - \sqrt{t} \cdot \tilde{O}(m)$, where m is the bitlength of X.[1]

In light of these facts, the approach of [Håstad et al. 1999] to constructing pseudorandom generators from one-way functions is the following three-step process:

1. **Computational Entropy Gap.** From an arbitrary one-way function, construct an efficiently samplable distribution X that has pseudoentropy at least $\text{H}(X) + \Delta$ for some $\Delta \geq 1/\text{poly}(n)$.
2. **Flattening.** Use flattening to obtain an efficiently samplable distribution whose pseudo-min-entropy is significantly larger than its (smoothed) max-entropy.
3. **Hashing.** Use randomness extraction and hashing (as in Lemma 25.1 and Lemma 25.2) to obtain a generator G whose output distribution is pseudorandom while it is generated using a short seed (and in particular has small max-entropy).

1. Here we are using the fact that we have defined computational indistinguishability with respect to nonuniform distinguishers, in order to ensure that $X \overset{c}{\equiv} X'$ implies that $X^t \overset{c}{\equiv} (X')^t$. The latter implication does not hold in general for uniform distinguishers Goldreich and Meyer [1998]. The implication does hold if X and X' can be sampled in polynomial time, but the constructions we will describe do not seem to have that property for X'. In [Haitner et al. 2013], this is remedied by a more complicated definition, where we require indistinguishability even by distinguishers that have an oracle for sampling from X', but where we also allow X' to depend on the distinguisher.

Unfortunately, the construction of [Håstad et al. 1999] ended up being much more complex and inefficient than this outline suggests. The main reasons are that in Step 1, (a) we do not know the real entropy $\mathrm{H}(X)$ of the samplable distribution X, and (b) the entropy gap Δ is quite small (and thus requires more repetitions for flattening to preserve the gap). In Section 25.6 we will see how to avoid these difficulties by using more refined notions of pseudoentropy.

Before proceeding, we define conditional versions of pseudoentropy that will be useful in later sections.

Definition 25.10 (Conditional pseudoentropy Hsiao et al. [2007]) Let (X, Y) be a pair of jointly distributed random variables of total length $\mathrm{poly}(n)$ for a security parameter n. We say that X has ***conditional pseudoentropy at least*** k given Y if there is a random variable X', jointly distributed with Y, such that:

1. $(X, Y) \overset{c}{\equiv} (X', Y)$.
2. $\mathrm{H}(X'|Y) \geq k$.

If Condition 2 is replaced with $\mathrm{H}_\infty(X'|Y) \geq k$, then we say that X has ***pseudo-min-entropy at least*** k given Y.

Similarly to the unconditional versions, conditional pseudoentropy supports flattening and randomness extraction by efficiently computable extractors.

25.5 One-Way Permutations to Pseudorandom Generators

In this section, we present the classic construction of pseudorandom generators from one-way *permutations* using the language of computational entropy. Specifically, we will prove the following theorem:

Theorem 25.2 (PRGs from OWPs Blum and Micali [1984], Yao [1982], Goldreich and Levin [1989]) If there exists a one-way permutation $f : \{0, 1\}^n \to \{0, 1\}^n$, then there exists a pseudorandom generator $G^f : \{0, 1\}^\ell \to \{0, 1\}^m$. Moreover, G^f makes $q = 1$ query to f and has seed length $\ell = O(n)$.

Note that this construction is extremely efficient, with only one query to the one-way function and a linear seed length.

Our presentation of the proof of Theorem 25.2 will use more complex concepts than the traditional presentation, in order to set the stage for Section 25.6, where we handle general one-way functions. The construction and security reduction implicit in the proof are actually the same as in the traditional presentation; they are just described using different language.

The first step of the proof is to observe that the definition of one-wayness can be directly related to a computational analogue of guessing entropy, as defined in Lemma 25.3, simply by restricting the guesser A to be efficient:

Definition 25.11 (Guessing pseudoentropy Hsiao et al. [2007]) Let X and Y be jointly distributed random variables of total length $\mathrm{poly}(n)$ for a security parameter n. We say that X has ***guessing pseudoentropy***[2] ***at least*** k given Y if for some $s(n) = n^{\omega(1)}$ and all nonuniform algorithms A running in time $s(n)$, we have

$$\Pr[A(Y) = X] \leq 2^{-k},$$

where the probability is taken over (X, Y) and the coin tosses of A.

If we take $Y = f(X)$ for a one-way function f, then the one-wayness of f implies that the above definition is satisfied for $2^{-k} = n^{-\omega(1)}$:

Lemma 25.7 If $f : \{0, 1\}^n \to \{0, 1\}^n$ is a one-way function, and X is uniformly distributed in $\{0, 1\}^n$, then X has guessing pseudoentropy $\omega(\log n)$ given $f(X)$.

Recall that guessing entropy is equal to average min-entropy in the information-theoretic setting (Lemma 25.3). In the computational setting, however, they are not equivalent; that is, guessing pseudoentropy is *not* in general equal to pseudo-min-entropy. Indeed, if f is a one-to-one one-way function, then X has *negligible* pseudo-min-entropy given $f(X)$, since for every X' such that $\mathrm{H}_\infty(X'|f(X))$ is nonnegligible, the efficient test $T(x, y)$ that outputs 1 iff $y = f(x)$ distinguishes $(X, f(X))$ from $(X', f(X))$.

Nevertheless, guessing pseudoentropy does support randomness extraction, not by arbitrary extractors, but ones meeting the following definition, which requires that the extractor is efficiently "list-decodable," in the sense that any test T that distinguishes the output of the extractor (on a fixed but unknown source element x) from uniform can be used to efficiently describe a list of at most 2^k elements that includes x. We will allow this list-decoding to be probabilistic and require it to succeed with some constant probability over its randomness r. Rather than asking the decoder to explicitly write down all 2^k elements of the list, we will index into the list by strings z of length k provided as input to the decoder.

2. This was called "unpredictability entropy" by Hsiao et al. [2007], but we use the term *guessing pseudoentropy* to highlight its relationship with *guessing entropy* (which happens to equal average min-entropy).

Definition 25.12 List-decodable extractors Trevisan [2001], Ta-Shma and Zuckerman [2004], Barak et al. [2003], Vadhan [2012].[3] A function $\mathrm{Ext} : \{0, 1\}^n \times \{0, 1\}^d \to \{0, 1\}^m$ is a (t, k, ε)-***list-decodable extractor*** if there is a nonuniform time t oracle algorithm $\mathrm{Dec} : \{0, 1\}^k \times \{0, 1\}^\ell \to \{0, 1\}^n$ such that the following holds: for every $x \in \{0, 1\}^n$ and $T : \{0, 1\}^d \times \{0, 1\}^m \to \{0, 1\}$ satisfying $\left|\Pr[T(U_d, \mathrm{Ext}(x, U_d)) = 1] - \Pr[T(U_d, U_m) = 1]\right| > \varepsilon$, we have

$$\Pr_{r \xleftarrow{R} \{0,1\}^\ell}\left[\exists z \in \{0, 1\}^k \;\; \mathrm{Dec}^T(z, r) = x\right] \geq \frac{1}{2}.$$

It is known that if we remove the time bound on the decoder t (i.e., set $t = \infty$), then list-decodable extractors are equivalent to standard (k, ε) randomness extractors up to an additive $\log(1/\varepsilon) + O(1)$ change in the min-entropy k and a constant factor in the error parameter ε. (See Vadhan [2012, Props. 6.23 and 7.72].)

Our reason for considering list-decodable extractors is that they extract pseudorandom bits from guessing pseudoentropy:

Lemma 25.8 (Extraction from guessing pseudoentropy Ta-Shma and Zuckerman [2004], Hsiao et al. [2007]) Let X and Y be jointly distributed random variables of total length $\mathrm{poly}(n)$ for a security parameter n, and where X has length m. Suppose X has guessing pseudoentropy at least k given Y, and that for every $t = n^{\omega(1)}$, there is an $\varepsilon = n^{-\omega(1)}$ such that $\mathrm{Ext} : \{0, 1\}^m \times \{0, 1\}^d \to \{0, 1\}^{m'}$ is a $(t, k - \log(3/\varepsilon), \varepsilon)$-list-decodable extractor. Then $(U_d, \mathrm{Ext}(X, U_d), Y) \overset{c}{\equiv} (U_d, U_m, Y)$.

Proof By the definition of guessing pseudoentropy, there is an $s = n^{\omega(1)}$ such that for every nonuniform A running in time s,

$$\Pr[A(Y) = X] \leq 2^{-k}.$$

Let $t = \sqrt{s} = n^{\omega(1)}$. By hypothesis, there is an $\varepsilon = n^{-\omega(1)}$ such that Ext is a (t, k', ε)-list-decodable extractor for $k' = k - \log(3/\varepsilon)$. Let $\mathrm{Dec} : \{0, 1\}^{k'} \times \{0, 1\}^\ell \to \{0, 1\}^n$ be as guaranteed by Definition 25.12.

We will show that no nonuniform time t algorithm T can distinguish $(U_d, \mathrm{Ext}(X, U_d), Y)$ and (U_d, U_m, Y) with advantage greater than 2ε. Suppose for contradiction that there is a nonuniform time t algorithm T such that

3. This is a variant of definitions that appear in the literature under different names, such as "reconstructive extractors" and "black-box pseudorandom generator constructions." In particular the definition in Vadhan [2012] of "black-box pseudorandom generator constructions" amounts to a definition of *locally* list-decodable extractors, where we only measure the time complexity of computing any one bit of the source string x, rather than all n bits at once.

$$\left|\Pr[T(U_d, \mathrm{Ext}(X, U_d), Y) = 1] - \Pr[T(U_d, U_m, Y) = 1]\right| > 2\varepsilon.$$

Then with probability at least ε over $(x, y) \overset{R}{\leftarrow} (X, Y)$, we have

$$\left|\Pr[T(U_d, \mathrm{Ext}(x, U_d), y) = 1] - \Pr[T(U_d, U_m, y) = 1]\right| > \varepsilon.$$

When this event occurs, we have

$$\Pr_{r \overset{R}{\leftarrow} \{0,1\}^\ell}\left[\exists z \in \{0, 1\}^{k'} \ \mathrm{Dec}^{T(\cdot,\cdot,y)}(z, r) = x\right] \geq \frac{1}{2}.$$

Therefore, if we define $A(y) = \mathrm{Dec}^{T(\cdot,\cdot,y)}(Z, R)$, where Z and R are both chosen uniformly at random, we have

$$\Pr[A(Y) = X] \geq \varepsilon \cdot \frac{1}{2} \cdot 2^{-k'} > 2^{-k}.$$

Moreover, being obtained from the time t algorithm Dec with an oracle T that is also a time t algorithm, A is a nonuniform algorithm running in time at most $t^2 = s$. This contradicts the guessing pseudoentropy of X given Y. ■

One of the many interpretations of the celebrated Goldreich–Levin Hardcore Bit Theorem is as providing a list-decodable extractor.

Theorem 25.3 (GL extractor Goldreich and Levin [1989]) For every $\varepsilon > 0$, the function $\mathrm{Ext}(x, r) = \left(\sum_i x_i r_i\right) \bmod 2$ is a $(\mathrm{poly}(n, 1/\varepsilon), 2\log(1/\varepsilon) + O(1), \varepsilon)$-list-decodable extractor.

Note that for any $k = \omega(\log n)$, the GL extractor satisfies the conditions of Lemma 25.8. Indeed, for any $t = n^{\omega(1)}$, if we set $\varepsilon = \max\{1/t^{1/c}, k/4\}$ for a large enough constant c, then Theorem 25.3 ensures that Ext is a $(t, k - \log(3/\varepsilon), \varepsilon)$-list-decodable extractor.

We now can prove Theorem 25.2, constructing a pseudorandom generator from any one-way permutation.

Proof (of Theorem 25.2) Let $f : \{0, 1\}^n \to \{0, 1\}^n$ be a one-way permutation, and let $\mathrm{Ext}(x, r) = \left(\sum_i x_i r_i\right) \bmod 2$. Define

$$G^f(x, r) = (r, \mathrm{Ext}(x, r), f(x)).$$

Note that G^f is polynomial-time computable with one query to f, has seed length $\ell = 2n$, and has output length $m = 2n + 1$.

All that remains is to prove the pseudorandomness of $G^f(U_\ell)$. Let X and R be random variables uniformly distributed in $\{0, 1\}^n$, set $Y = f(X)$. By Lemma 25.7, X

has guessing pseudoentropy $\omega(\log n)$ given Y. By Theorem 25.3 and Lemma 25.8, we have

$$G^f(X, R) \equiv (R, \mathrm{Ext}(X, R), Y) \overset{c}{\equiv} (R, U_1, Y) \equiv U_{2n+1}. \quad ■$$

Before moving on to the case of general one-way functions, we revisit the relationship between guessing pseudoentropy and ordinary pseudo-min-entropy. As noted above, the example $Y = f(X)$ for a one-to-one one-way function f shows that X having noticeable guessing pseudoentropy given Y does not in general imply that X has noticeable pseudo-min-entropy given Y. However, it turns out that this implication does hold when X is *short*:

Theorem 25.4 (Guessing pseudoentropy vs. pseudo-min-entropy Zheng [2014], Skórski et al. [2015]) Let (X, Y) be jointly distributed random variables, where X is distributed over strings of length $\ell = O(\log n)$ and Y is distributed over strings of length poly(n), for a security parameter n. Then for every $k \in [0, \ell]$, the following are equivalent:

1. There is a negligible $\varepsilon = \varepsilon(n)$, such that X has guessing pseudoentropy at least $k - \varepsilon$ given Y.
2. There is a negligible $\varepsilon = \varepsilon(n)$ such that X has pseudo-min-entropy at least $k - \varepsilon$ given Y.[4]

As discussed by Zheng [2014], the case of Boolean X (i.e., $\ell = 1$) amounts to a reinterpretation of (tight) versions of Impagliazzo's Hardcore Theorem [Impagliazzo 1995, Klivans and Servedio 2003, Barak et al. 2009, Sudan et al. 2001]. We also remark that the version of Theorem 25.4 by Skórski, Golovnev, and Pietrzak [Skórski et al. 2015] relaxes the condition that X is short (i.e., $\ell = O(\log n)$) to the pseudoentropy deficiency being small (i.e., $\ell - k = O(\log n)$).

25.6 One-Way Functions to Pseudorandom Generators

We now turn to constructing pseudorandom generators from arbitrary one-way functions. Specifically, we will sketch the most efficient construction to date:

Theorem 25.5 (Improved PRGs from OWFs Haitner et al. [2013], Vadhan and Zheng [2012]) If there exists a one-way function $f : \{0, 1\}^n \to \{0, 1\}^n$, then there exists a pseudo-

4. Actually, we can replace Item 2 with the statement that X has pseudo-min-entropy at least k given Y, with no negligible loss, by exploiting the slackness afforded by indistinguishability. Indeed, suppose $(X, Y) \overset{c}{\equiv} (X', Y)$ where $\mathrm{H}_\infty(X'|Y) \geq k - \varepsilon$. It can be shown that (X', Y) is *statistically* indistinguishable from some (X'', Y) such that $\mathrm{H}_\infty(X''|Y) \geq k$. Then (X, Y) is also computationally indistinguishable from (X'', Y), and hence X has pseudo-min-entropy at least k given Y.

random generator $G^f : \{0, 1\}^\ell \to \{0, 1\}^m$ with seed length $\ell = \tilde{O}(n^3)$ that makes $q = \tilde{O}(n^3)$ queries to f.

Pseudoentropy from One-Way Functions. Like the proof of Theorem 25.2 given above, we will begin the proof of Theorem 25.5 by looking for some form of pseudoentropy in an arbitrary one-way function f. Note that the fact that X has guessing pseudoentropy $\omega(\log n)$ given $f(X)$ (Lemma 25.7) holds for every one-way function, regardless of whether or not it is one-to-one. However, when f is many-to-one, this fact may hold for trivial information-theoretic reasons. Indeed, consider any function f that ignores the first half of its input. Then X has average min-entropy at least $n/2$ given $f(X)$, so in particular has guessing pseudoentropy at least $n/2 = \omega(\log n)$ given $f(X)$, regardless of the one-wayness of f.

Thus, we need to replace guessing pseudoentropy with a notion that captures the *gap* between the computational and information-theoretic hardness in X given $f(X)$. To do so, we need to exploit the fact that one-wayness guarantees that it is hard to find *any* preimage of $f(X)$, something that is not captured by guessing pseudoentropy. We will do this by using a computational analogue of KL divergence (a.k.a. relative entropy). We begin with the information-theoretic definition.

Definition 25.13 (KL divergence) Let A and A' be two discrete random variables. The *Kullback–Leibler (KL) divergence from A to A'* is

$$\mathrm{KL}\left(A \| A'\right) = \mathrm{E}_{a \xleftarrow{R} A}\left[\log\left(\frac{\Pr[A = a]}{\Pr[A' = a]}\right)\right].$$

It can be shown that $\mathrm{KL}\left(A \| A'\right) \geq 0$, with equality iff A and A' are identically distributed. Thus KL divergence can be thought of as a measure of "distance" between probability distributions, but note that it is not symmetric and does not satisfy the triangle inequality. Also note that $\mathrm{KL}\left(A \| A'\right)$ is infinite if (and only if) $\mathrm{Supp}(A) \nsubseteq \mathrm{Supp}(A')$.

For intuition about KL divergence, it is useful to consider the case of flat distributions (where A and A' are uniform on their supports). Then, if $\mathrm{Supp}(A) \subseteq \mathrm{Supp}(A')$, $\mathrm{KL}\left(A \| A'\right) = \log(|\mathrm{Supp}(A')|/|\mathrm{Supp}(A)|)$, so $\mathrm{KL}(A||A')$ measures how densely A is contained in A'. More generally, if A' is flat and A is an arbitrary random variable such that $\mathrm{Supp}(A) \subseteq \mathrm{Supp}(A')$, then $\mathrm{KL}\left(A \| A'\right) = \log|\mathrm{Supp}(A')| - \mathrm{H}(A) = \mathrm{H}(A') - \mathrm{H}(A)$.

We will also refer to a conditional version of KL divergence.

Definition 25.14 (Conditional KL divergence) Let (A, B) and (A', B') be two pairs of discrete random variables. The *Kullback–Leibler (KL) divergence from $A|B$ to $A'|B'$* is

$$\begin{aligned}\mathrm{KL}\left(A|B\|A'|B'\right) &= \mathrm{E}_{b\overset{R}{\leftarrow}B}\left[\mathrm{KL}\left(A|_{B=b}\|A'|_{B'=b}\right)\right]\\ &= \mathrm{KL}\left((A,B)\|(A',B')\right) - \mathrm{KL}\left(B\|B'\right).\end{aligned}$$

Note that the dependence of $\mathrm{KL}\left(A|B\|A'|B'\right)$ on (A', B') involves only the family of conditional probability distributions $\{A'|_{B'=b}\}$. In particular, it does not depend on the marginal distribution of B'.

The computational analogue of KL divergence we will use is the following:

Definition 25.15 (Conditional KL-hardness Vadhan and Zheng [2012]) Let (X, Y) be a pair of jointly distributed random variables of total length $\mathrm{poly}(n)$, where n is the security parameter. We say that X is ***Δ-KL-hard given*** Y iff for some $s(n) = n^{\omega(1)}$ and all nonuniform algorithms A running in time $s(n)$, we have

$$\mathrm{KL}\left(X|Y\|A(Y)|Y\right) \geq \Delta.$$

Equivalently, we require

$$\mathrm{KL}\left((X, Y)\|(A(Y), Y)\right) \geq \Delta.$$

The goal of the adversary A is to minimize the divergence $\mathrm{KL}\left(X|Y\|A(Y)|Y\right)$. To make the divergence small, $A(y)$ should output a distribution that is as close as possible to the conditional distribution $X|_{Y=y}$. That is, the distribution $A(y)$ should contain the distribution $X|_{Y=y}$ as tightly as possible.

A computationally unbounded adversary A can achieve zero divergence by having $A(y)$ be distributed exactly according to the conditional distribution $X|_{Y=y}$. Therefore, X being Δ-KL-hard-to-sample given Y for a nonzero Δ is a statement purely about computational hardness, not information-theoretic hardness. This is in contrast to guessing pseudoentropy, which can be large for purely information-theoretic reasons (as discussed earlier).

We can still show that an arbitrary one-way function gives us KL hardness:

Lemma 25.9 (KL-hardness from one-way functions Vadhan and Zheng [2012]) If $f : \{0, 1\}^n \to \{0, 1\}^n$ is a one-way function and X is uniformly distributed in $\{0, 1\}^n$, then X is $\omega(\log n)$-KL-hard given $f(X)$.

Proof Sketch Like statistical difference, KL divergence has the property it cannot be increased by applying a function. That is, for all functions T and random variables W and Z, $\mathrm{KL}\left(T(W)\|T(Z)\right) \leq \mathrm{KL}\left(W\|Z\right)$. This fact is known as the *data-processing inequality* for KL divergence. We will apply this inequality with the test $T(x, y)$ that outputs 1 if $y = f(x)$ and 0 otherwise. Specifically, for every adversary A running in time

$s(n) = n^{\omega(1)}$, we have

$$\begin{aligned}
&\mathrm{KL}\left((X, f(X)) \| (A(f(X)), f(X))\right) \\
&\quad \geq \mathrm{KL}\left(T(X, f(X)) \| T(A(f(X)), f(X))\right) && \text{(data-processing inequality)} \\
&\quad = \log\left(\frac{1}{\Pr[T(A(f(X)), f(X)) = 1]}\right) && (\Pr[T(X, f(X)) = 1] = 1) \\
&\quad = \log\left(\frac{1}{\Pr[A(f(X)) \in f^{-1}(f(X))]}\right) && \text{(def of } T) \\
&\quad = \log(n^{\omega(1)}) && \text{(one-wayness of } f) \\
&\quad = \omega(\log n).
\end{aligned}$$

■

Similarly to Theorem 25.4, we can also relate KL-hardness to pseudoentropy when X is short:

Theorem 25.6 (KL-hardness vs. pseudoentropy Vadhan and Zheng [2012]) Let (X, Y) be jointly distributed random variables, where X is distributed over strings of length $\ell = O(\log n)$ and Y is distributed over strings of length $\mathrm{poly}(n)$, for a security parameter n. Then for every $\Delta \in [0, \ell - \mathrm{H}(X|Y)]$, the following are equivalent:

1. There is a negligible $\varepsilon = \varepsilon(n)$ such that X is $(\Delta - \varepsilon)$-KL-hard given Y.
2. There is a negligible $\varepsilon = \varepsilon(n)$ such that X has pseudoentropy at least $\mathrm{H}(X|Y) + \Delta - \varepsilon$ given Y.[5]

Note that, as we desired, the KL-hardness quantifies the *gap* between the pseudoentropy and the real entropy $\mathrm{H}(X|Y)$.

However, we cannot directly combine Theorem 25.6 and Lemma 25.9, since the input X to a one-way function is not short. Fortunately, KL-hardness is preserved if we break X up into short blocks:

Lemma 25.10 (Blockwise KL-hardness [Vadhan and Zheng 2012]) Let (X, Y) be a pair of jointly distributed random variables of total length $\mathrm{poly}(n)$, where n is the security parameter, and let $X = (X_1, \ldots, X_m)$ be a partition of X into blocks. If X is Δ-KL-hard given Y, then for I uniformly distributed in $\{1, \ldots, m\}$ X_I is (Δ/m)-KL-hard given $(Y, X_1, \ldots, X_{I-1})$.

Proof Sketch Suppose for contradiction that there is an efficient adversary A such that

$$\mathrm{KL}\left((Y, X_1, \ldots, X_I) \| (Y, X_1, \ldots, X_{I-1}, A(Y, X_1, \ldots, X_{I-1}))\right) < \Delta/m.$$

5. Similarly to Footnote 4, the negligible loss of ε in Item 2 can be removed.

That is, A samples one block X_I given Y and the previous blocks $X_1, \ldots, X_{I-1}$ with approximately the correct distribution. We now construct an adversary B that uses A iteratively to sample all of X given only Y. Specifically, $B(y)$ defined as follows:

- For $i = 1, \ldots, m$, let $x_i = A(y, x_1, \ldots, x_{i-1})$.
- Output $x = (x_1, \ldots, x_m)$.

Notice that if A achieves divergence zero—that is, $A(y, x_1, \ldots, x_{i-1})$ is always distributed exactly according to the conditional distribution $X_i|_{Y=y, X_1=x_1, \ldots, X_{i-1}=x_{i-1}}$—then B will also achieve divergence zero, i.e., $B(y)$ is always identically distributed to $X|_{Y=y}$. More generally, it can be shown that the divergence achieved by B equals the sum of the divergences achieved by A over the m blocks. That is,

$$\begin{aligned}
&\mathrm{KL}\,(X|Y \| B(Y)|Y)\\
&\quad= \sum_{i=1}^{m} \mathrm{KL}\left(X_i|(Y, X_1, \ldots, X_{i-1}) \| A(Y, X_1, \ldots, X_{i-1})|(Y, X_1, \ldots, X_{i-1})\right),\\
&\quad= m \cdot \mathrm{KL}\left(X_I|(Y, X_1, \ldots, X_{I-1}) \| A(Y, X_1, \ldots, X_{I-1})|(Y, X_1, \ldots, X_{I-1})\right),\\
&\quad< \Delta,
\end{aligned}$$

contradicting the KL hardness of X given Y. ■

Combining Lemma 25.10 and Theorem 25.6, we see that if X is Δ-KL-hard given Y and we partition X into m short blocks, then, on average, those blocks will have pseudoentropy larger than their real entropy by Δ/m (given the previous blocks and Y). The latter conclusion can be reinterpreted using the following blockwise notion of pseudoentropy:

Definition 25.16 (Next-block pseudoentropy Haitner et al. [2013]) Let $X = (X_0, X_1, \ldots, X_m)$ be a sequence of random variables distributed on strings of total length $\mathrm{poly}(n)$ for a security parameter n. We say that X has *next-block pseudoentropy* at least k if there is a sequence of random variables $(X'_0, X'_1, \ldots, X'_m)$, jointly distributed with X, such that:

1. For each $i = 0, \ldots, m$, $(X_0, X_1, \ldots, X_{i-1}, X_i) \overset{c}{\equiv} (X_0, X_1, \ldots, X_{i-1}, X'_i)$.
2. $\sum_{i=0}^{m} \mathrm{H}(X'_i|X_0, \ldots, X_{i-1}) \geq k$.

That is, to an "online" adversary that observes the random variables $(X_0, \ldots, X_m)$ in sequence, at each step the next block X_i looks like a "higher entropy" random variable X'_i. For comparison, consider the notion of next-bit *pseudorandomness*, where each of the blocks is of length 1 and $(X_1, \ldots, X_{i-1}, X_i) \overset{c}{\equiv} (X_1, \ldots, X_{i-1}, X'_i)$ where X'_i is a uniformly random bit independent of $(X_1, \ldots,$

X_{i-1}). As asserted by Yao, next-bit pseudorandomness is equivalent to both the notion of next-bit unpredictability of Blum and Micali [1984] as well as to pseudorandomness of the entire sequence as in Definition 25.7. Next-block pseudoentropy can be thought of as a quantitative generalization of this classic notion, but, importantly, it is *not* generally equivalent to pseudoentropy of the entire sequence as demonstrated by the following theorem and discussion:

Theorem 25.7 (Next-block pseudoentropy from OWFs Vadhan and Zheng [2012]) Let $f : \{0, 1\}^n \to \{0, 1\}^n$ be a one-way function, let X be uniformly distributed in $\{0, 1\}^m$, and let $X = (X_1, \ldots, X_m)$ be a partition of X into blocks of length $O(\log n)$. (For example, we can set $m = n$ and set X_i to be the i'th bit of X.) Then the sequence $(f(X), X_1, \ldots, X_m)$ has next-block pseudoentropy at least $n + \omega(\log n)$.

As discussed earlier, the global pseudoentropy of the random variable $(f(X), X)$ is at most $n + n^{-\omega(1)}$, since the test $T(y, x)$ that checks whether $y = f(x)$ distinguishes $(f(X), X)$ from every distribution of entropy noticeably more than n. Theorem follows combining Lemma 25.9, Lemma 25.10, and Theorem 25.6.

Notice that the amount of next-block pseudoentropy in $(f(X), X)$ is $\omega(\log n)$ bits larger than the number of random bits we need to generate it (choosing a uniformly random X). Thus, if we can extract this pseudoentropy to produce pseudorandomness, we will have a pseudorandom generator. Unfortunately, Theorem 25.7 only guarantees pseudoentropy in the Shannon sense, whereas we need (pseudo-)min-entropy to extract (Lemma 25.6). Thus, we first need to apply flattening (Lemma 25.5).

Flattening Pseudoentropy. By Theorem 25.7, there are real numbers $k_0, k_1, \ldots, k_m \geq 0$ such that if we let $X_0 = f(X)$ and $X = (X_1, \ldots, X_m)$, we have:

1. $\sum_{i=0}^{m} k_i = n + \omega(\log n)$.
2. X_i has pseudoentropy at least k_i given $X_0, \ldots, X_{i-1}$ for $i = 0, \ldots, m$.

To flatten, we take t independent inputs $X^{(1)}, \ldots, X^{(t)}$ sampled uniformly from $\{0, 1\}^n$ for the one-way function f, define blocks for each by setting $X_0^{(i)} = f(X^{(i)})$ and $(X_1^{(i)}, \ldots, X_m^{(t)}) = X^{(i)}$, and create larger blocks $\tilde{X}_0, \ldots, \tilde{X}_m$ as follows:

- $\tilde{X}_0 = (f(X^{(1)}), f(X^{(2)}), \ldots, f(X^{(t)}))$.
- $\tilde{X}_i = (X_i^{(1)}, X_i^{(2)}, \ldots, X_i^{(t)})$ for $i = 1, \ldots, m$.

Then using Lemma 25.5 (and its generalization to conditional entropy) with $\varepsilon = n^{-\log n}$, it can be shown that for each i, the block $\tilde{X}_i$ has pseudo-min-entropy at least $\tilde{k}_i = t \cdot k_i - O(\sqrt{t} \cdot \log n \cdot \ell_i)$, where ℓ_i is the bit-length of the ith block.

For $t = n^2$, it can be checked that $\sum_i \tilde{k}_i = t \cdot (n + \omega(\log n))$, so the pseudo-min-entropy in the blocks $\tilde{X}_i$ is (significantly) larger than the $t \cdot n$ bits used to generate them. Applying a randomness extractor to each of these larger blocks, we obtain the following pseudorandom generator.

A "Nonuniform" Pseudorandom Generator. The following construction requires knowledge of the entropy thresholds k_i, which may be hard to compute and thus are provided as nonuniform advice to the pseudorandom generator. Later we will see how to remove this nonuniformity.

The seed of our pseudorandom generator consists of the t independent inputs $x^{(1)}, \ldots, x^{(t)}$ to f, and descriptions of universal hash functions $h_0, \ldots, h_m$ where h_i has output length $\tilde{k}_i - \omega(\log n)$, and the output is

$$G^f(x^{(1)}, \ldots, x^{(t)}, h_0, \ldots, h_m) = (h_0, \ldots, h_m, h_0(\tilde{x}_0), \ldots, h_m(\tilde{x}_m)).$$

Pictorially:

$$\begin{array}{c|c|c|c|c|c}
 & \tilde{x}_0 & \tilde{x}_1 & \cdots & \tilde{x}_m & \\
 & \| & \| & & \| & \\
\hline
x^{(1)} = & f(x^{(1)}) & x_1^{(1)} & \cdots & x_m^{(1)} & \\
x^{(2)} = & f(x^{(2)}) & x_1^{(2)} & \cdots & x_m^{(2)} & \\
\vdots & \vdots & \vdots & \vdots & \vdots & \\
x^{(t)} = & f(x^{(t)}) & x_1^{(t)} & \cdots & x_m^{(t)} & \\
\hline
\uparrow & \downarrow & \downarrow & \cdots & \downarrow & \\
\text{seed} \rightarrow & h_0 & h_1 & \cdots & h_m & \rightarrow \text{output} \\
 & \downarrow & \downarrow & \cdots & \downarrow & \\
 & h_0(\tilde{x}_0) & h_1(\tilde{x}_1) & \cdots & h_m(\tilde{x}_m) & \rightarrow \text{output}
\end{array}$$

It can be proven, following the arguments sketched above, that the output of this generator is indeed pseudorandom and longer than its seed length.

Entropy Equalization. We address the nonuniformity issue above by the following "entropy equalization" technique, which converts any next-block pseudoentropy generator into one where every block has guaranteed to have at least the average amount of pseudoentropy of the blocks in the original generator. It works by concatenating many independent samples of the next-block-pseudoentropy generator, but left-shifted by a random offset from $\{0, 1, \ldots, m\}$, so that each block of the new generator has equal probability of being each of the $m + 1$ blocks of the original generator. We do not use a cyclic shift, but rather drop appropriate parts of the first and last blocks.

Lemma 25.11 (Entropy equalization Haitner et al. [2009b], Haitner et al. [2013]) Let $X = (X_0, \ldots, X_m)$ be a random variable distributed on strings of length $\text{poly}(n)$, where n is a security parameter. Suppose X has next-block pseudoentropy at least k. For a parameter $u \in \mathbb{N}$, consider the random variable $\hat{X}$ defined as follows:

1. Let $X^{(1)}, \ldots, X^{(u)}$ be u independent samples of X, with blocks $X^{(i)} = (X_0^{(i)}, \ldots, X_m^{(i)})$.
2. Choose $J \xleftarrow{R} \{0, \ldots, m\}$.
3. Output

$$\begin{aligned}\hat{X} &= (\hat{X}_0, \hat{X}_1, \ldots, \hat{X}_{(u-1)\cdot(m+1)}) \\ &\stackrel{\text{def}}{=} (J, X_J^{(1)}, X_{J+1}^{(1)}, \ldots, X_m^{(1)}, X_0^{(2)}, \ldots, X_m^{(2)}, \ldots X_0^{(u-1)}, \ldots, \\ &\qquad X_m^{(u-1)}, X_0^{(u)}, \ldots, X_{J-1}^{(u)}).\end{aligned}$$

That is, $\hat{X}_0 = J$ and $\hat{X}_i = X_{(J+i+m) \bmod (m+1)}^{(\lfloor (J+i+m)/(m+1) \rfloor)}$ for $i = 1, \ldots, (u-1)\cdot(m+1)$.

Then for every $i = 1, \ldots, (u-1)\cdot(m+1)$, $\hat{X}_i$ has pseudoentropy at least $k/(m+1)$ given $\hat{X}_0, \ldots, \hat{X}_{i-1}$.

As stated above, the left-shifting is not cyclic; we drop J blocks of $X^{(1)}$ and $m + 1 - J$ blocks of $X^{(u)}$. Intuitively, $\hat{X}_i$ has pseudoentropy at least $k/(m+1)$ given the previous blocks because $\hat{X}_i$ is a copy of $X_{(J+i+m) \bmod (m+1)}$ and $(J + i + m) \bmod (m+1)$ is uniformly distributed in $\{0, \ldots, m\}$. Notice that the total next-block pseudoentropy guaranteed in the blocks $\hat{X}_i$ for $i > 1$ is $((t-1)\cdot(m+1))\cdot(k/(m+1)) = (t-1)\cdot k$. We generated $\hat{X}$ using t copies of X, which had $t \cdot k$ bits of next-block pseudoentropy, but we lost one copy's worth of pseudoentropy by discarding a prefix of the first copy and a suffix of the last copy.

Applying this to the next-block pseudoentropy generator of Theorem 25.7, with $k = n + \omega(\log n)$, we can take $u = n/\log n$ and have

$$(u-1)\cdot k = u \cdot (n + \omega(\log n)) - k = u \cdot (n + \omega(\log n)),$$

so we still have much more pseudoentropy than the $u \cdot n + \log u$ bits used to generate $\hat{X}$.

Applying the flattening and extraction procedure to this entropy-equalized next-block pseudoentropy generator (rather than to the one of Theorem 25.7), we obtain a uniformly computable pseudorandom generator that makes $q = u \cdot t = O(n^3)$

queries to the one-way functions (with a factor of $t = O(n^2)$ coming from flattening and a factor of $u = O(n)$ from entropy equalization), and has seed length $O(q \cdot n) = O(n^4)$.

To save an extra factor of n in the seed length as claimed in Theorem 25.5, the idea is to show that the repetitions used for entropy equalization need not be independent, and instead randomness can be (adaptively) recycled in a way that is similar to the length expansion for pseudorandom generators. We refer to Vadhan and Zheng [2012] for more details.

25.7 One-Way Functions to Statistically Hiding Commitments

In this section, we describe how another form of computational entropy, *inaccessible entropy*, is used to construct statistically hiding commitment schemes from one-way functions. In doing so, we will highlight the duality between the construction and notions used below and those that were used above for constructing pseudorandom generators.

Commitment Schemes. Recall that a ***commitment scheme*** is a two-party protocol between a ***sender*** S and a ***receiver*** R. The protocol consists of two phases. In the ***commit phase***, the sender takes as input a message m of length $\text{poly}(n)$, in addition to both parties receiving the security parameter n. In the ***reveal phase***, the sender reveals the message m to the receiver and "proves" that m is the message to which it committed in the first phase, after which the receiver accepts or rejects. Without loss of generality, the sender's proof can consist of the coin tosses r she used in the commit phase, and the receiver simply checks that the transcript of the commit phase is consistent with the behavior of the sender algorithm $S(m; r)$ on message m and coin tosses r.

A commitment scheme has two security requirements. Informally, the ***hiding*** property requires that the receiver should learn nothing about the message m during the commit phase. The ***binding*** property requires that after the commit phase, there should be a unique message m that the sender can successfully reveal. Typically, one of these two security properties is statistical (with security against computationally unbounded adversaries), while the other is computational.

Statistically binding commitments can be constructed from any pseudorandom generator [Naor 1991], and hence from any one-way function by Theorem 25.1. Thus, our focus in this section is on the analogous result for statistically hiding commitments:

Theorem 25.8 (Statistically hiding commitments from OWF Haitner et al. [2009a]) If there exists a one-way function, then there exists a statistically hiding commitment scheme.

The original proof of Theorem 25.8 was even more complex than the original proof of Theorem 25.1. Haitner, Reingold, Vadhan, and Wee [Haitner et al. 2009b] gave a much simpler and more conceptual proof using a new computational notion of entropy, called *inaccessible entropy*. That construction actually predated and inspired the more efficient construction of pseudorandom generators from one-way functions given in Section 25.6.

Commitment Schemes and Computational Entropy. We begin by explaining, at an intuitive level, the relationship between commitment schemes and notions of computational entropy.

A *statistically binding* commitment scheme is very related to pseudorandomness and pseudoentropy. As mentioned above, Naor [1991] exhibited a very efficient construction of statistically binding commitments from any pseudorandom generator. Conversely, consider running a statistically binding commitment protocol on a uniformly random message M, and let T be the transcript of the commit phase. Then the statistical binding property implies that M has negligible real entropy given T (since with all but negligible probability over $t \leftarrow T$, there should be only one message m in the support of $M|_{T=t}$). On the other hand, the computational hiding property implies that M is pseudorandom given T (i.e., $(M, T) \overset{c}{\equiv} (U, T)$, where U is a uniformly random message independent of M and T). So the pseudoentropy of M given T is much higher than the real entropy of M given T.

Let us now consider the case of a *statistically hiding* commitment scheme. The statistical hiding property implies that M is statistically close to uniform given T (i.e., $(M, T) \overset{s}{\equiv} (U, T)$). On the other hand, the computational binding property says that, from the perspective of a polynomial-time sender, M is effectively determined by T. That is, although M has a lot of real entropy given T, a computationally bounded algorithm cannot "access" this entropy. This motivates the following definition of (*next-block*) *accessible entropy*, which should be thought of as "dual" to (next-block) pseudoentropy (Definition 25.16):

Definition 25.17 (Next-block accessible entropy Haitner et al. [2009b]) Let n be a security parameter, and $Y = (Y_1, \ldots, Y_m)$ be a random variable distributed on strings of length poly(n). We say that Y has *next-block accessible entropy at most k* if the following holds for some $s(n) = n^{\omega(1)}$.

Let $\tilde{G}$ be any nonuniform, probabilistic algorithm running in time $s(n)$ that takes a sequence of uniformly random strings $\tilde{R} = (\tilde{R}_1, \ldots, \tilde{R}_m)$ and outputs a sequence $\tilde{Y} = (\tilde{Y}_1, \ldots, \tilde{Y}_m)$ in an "online fashion" by which we mean that $\tilde{Y}_i = \tilde{G}(\tilde{R}_1, \ldots, \tilde{R}_i)$ depends on only the first i random strings of $\tilde{G}$ for $i = 1, \ldots, m$. Suppose further that $\mathrm{Supp}(\tilde{Y}) \subseteq \mathrm{Supp}(Y)$.

Then we require

$$\sum_{i=1}^{m} \mathrm{H}(\tilde{Y}_i | \tilde{R}_1, \ldots, \tilde{R}_{i-1}) \leq k.$$

For intuition, think of each individual block Y_i as corresponding to a message being committed to in a statistically hiding commitment scheme, and the prefix $Y_{<i} = (Y_1, \ldots, Y_{i-1})$ as the transcript of a commit phase for Y_i. The adversary $\tilde{G}$ is analogous to a sender trying to break the computational binding property of the commitment scheme. $\tilde{G}$ is trying to generate a message $\tilde{Y}_i$ with as much entropy as possible, conditioned on its internal state after the commit phase, which is represented by its prior coin tosses $\tilde{R}_1, \ldots, \tilde{R}_{i-1}$. The condition that $\mathrm{Supp}(\tilde{Y}) \subseteq \mathrm{Supp}(Y)$ is analogous to the fact that the reveal phase of a commitment scheme demands that the message revealed is consistent with the transcript of the commit phase. Indeed, the security properties of a statistically hiding commitment scheme can be captured by using a generalization of the definition of accessible entropy to messages in interactive protocols [Haitner et al. 2009b].

(Next-block) accessible entropy differs from (next-block) pseudoentropy in two ways:

1. Accessible entropy is useful as an *upper* bound on computational entropy, and is interesting when it is *smaller* than the real entropy $\mathrm{H}(Y)$. We refer to the gap $\mathrm{H}(Y) - k$ as the *inaccessible entropy* of Y.
2. The accessible entropy adversary $\tilde{G}$ is trying to *generate* the random variables Y_i conditioned on the history rather than recognize them. Note that we take the "history" to not only be the previous blocks $(\tilde{Y}_1, \ldots, \tilde{Y}_{i-1})$, but the coin tosses $(\tilde{R}_1, \ldots, \tilde{R}_{i-1})$ used by $\tilde{G}$ to generate those blocks. This ensures that the randomness we measure in $\tilde{Y}_i$ comes only from $\tilde{R}_i$, so $\tilde{G}$ really needs to operate in an online fashion. [6]

The proof of Theorem 25.8 begins by showing that every one-way function has next-block inaccessible entropy:

Theorem 25.9 (Inaccessible entropy from OWFs Haitner et al. [2009b]) Let $f : \{0,1\}^n \to \{0,1\}^n$ be a one-way function, let X be uniformly distributed in $\{0,1\}^n$, and let $(Y_1, \ldots, Y_m)$

6. If we had conditioned only on the prior output blocks $\tilde{Y}_1, \ldots, \tilde{Y}_{i-1}$, then an adversary that runs an honest sampling algorithm once for $Y = (Y_1, \ldots, Y_m)$ would achieve accessible entropy $\sum_i \mathrm{H}(\tilde{Y}_i|\tilde{Y}_1, \ldots, \tilde{Y}_{i-1}) = \sum_i \mathrm{H}(Y_i|Y_1, \ldots, Y_{i-1}) = \mathrm{H}(Y)$. Here the entire sequence is determined by $\tilde{R}_1$, the coin tosses for generating Y, but we can get nonzero entropy for blocks 2–m since $\tilde{Y}_1$ will not determine $\tilde{R}_1$ in general.

be a partition of $Y = f(X)$ into blocks of length $O(\log n)$. Then $(Y_1, \ldots, Y_m, X)$ has next-block accessible entropy at most $n - \omega(\log n)$.

Notice that this statement is similar to Theorem 25.7, except that it refers to accessible entropy rather than pseudoentropy, it asserts an upper bound rather than a lower bound on the computational entropy, and that it requires partitioning $f(X)$ rather than X into short blocks.

Given Theorem 25.9, the construction of statistically hiding commitments from one-way functions (Thm. 25.8) follows the same template as what we saw for pseudorandom generators in Section 25.6:

1. An "entropy equalization" step that converts $Y = (Y_1, Y_2, \ldots, Y_{m+1})$ into a random variable $\hat{Y} = (Y_0, Y_2, \ldots, Y_{\hat{m}})$ generator where (a lower bound on) the real entropy in each block conditioned on the prior blocks before it is known, and the total next-block accessible entropy is significantly smaller than the total real entropy. The construction is exactly the same as in Lemma 25.11.
2. A "flattening" step that converts the real Shannon entropy guarantees into real min-entropy. Specifically, after flattening each block will have high (smoothed) min-entropy, while the total next-block accessible entropy is significantly smaller than the total smoothed min-entropy. This construction is again exactly the same as what we saw for pseudorandom generators. Note that we do not claim that the accessible entropy gets converted into accessible *max*-entropy by flattening; the reason is that the adversarial generator need not behave independently across the repetitions of flattening.
3. A "hashing" step that converts the high min-entropy in each block to nearly uniform randomness, and turns the low accessible entropy into a weak binding property (uniquely determining the block with noticeable probability, similar in spirit to Lemma 25.2). The reason that the binding property is weak comes from the fact that we only have a bound on accessible Shannon entropy (as discussed above) and from the fact that an adversarial generator has freedom in how to spread the accessible entropy across the blocks. Moreover, in order to tolerate potentially malicious senders (as is required for binding), it is not enough to directly apply universal hashing, as the sender could then decide on the message/block $\tilde{Y}_i$ after seeing the hash function. Instead, we use (information-theoretic) "interactive hashing" [Naor et al. 1998, Ding et al. 2007], which is designed to address this issue. Constructing full-fledged statistically hiding commitments in this step also utilizes universal one-way

hash functions [Naor and Yung 1989], which can be constructed from one-way functions [Rompel 1990], as well some additional repetitions to amplify the weak binding property. Without universal one-way hash functions, we obtain a non-standard weak binding property, which nevertheless suffices for some applications, such as constructing statistical zero-knowledge arguments for all of **NP**.

For the proof of Theorem 25.9, we recommend the recent work of Agrawal, Chen, Horel, and Vadhan [Agrawal et al. 2019], which gives a new, more modular proof that uses a strengthening of KL-hardness (Definition 25.15), which further illuminates the duality between next-block pseudoentropy and next-block accessible entropy.

References

R. Agrawal, Y.-H. Chen, T. Horel, and S. Vadhan. 2019. Unifying computational entropies via Kullback-Leibler divergence. Technical Report 1902.11202 [cs.CR], arXiv. 694, 722

B. Barak, R. Shaltiel, and A. Wigderson. 2003. Computational analogues of entropy. In S. Arora, K. Jansen, J. D. P. Rolim, and A. Sahai, editors, *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques*, pp. 200–215. Springer. DOI: 10.1007/978-3-540-45198-3_18. 695, 708

B. Barak, M. Hardt, and S. Kale. 2009. The uniform hardcore lemma via approximate Bregman projections. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pp. 1193–1200. Society for Industrial and Applied Mathematics, Philadelphia. DOI: 10.1137/1.9781611973068.129. 710

M. Bellare. 2002. A note on negligible functions. *Journal of Cryptology*, 15(4): 271–284. DOI: 10.1007/s00145-002-0116-x. 701

M. Bellare, S. Micali, and R. Ostrovsky. 1990. Perfect zero-knowledge in constant rounds. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, STOC '90, pp. 482–493. ACM, New York. DOI: 10.1145/100216.100283. 696

C. H. Bennett, G. Brassard, and J.-M. Robert. 1988. Privacy amplification by public discussion. *SIAM Journal on Computing*, 17(2): 210–229. Special issue on cryptography. DOI: 10.1137/0217014. 697

M. Blum and S. Micali. 1984. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM Journal on Computing*, 13(4): 850–864. DOI: 10.1137/0213053. 693, 695, 702, 706, 715

G. Brassard, D. Chaum, and C. Crépeau. 1988. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2): 156–189. DOI: 10.1016/0022-0000(88)90005-0. 694

B. Chor and O. Goldreich. 1988. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computing*, 17(2): 230–261. DOI: 10.1137/0217015. 697

K.-M. Chung, Y. T. Kalai, F.-H. Liu, and R. Raz. 2011. Memory delegation. In *Proceedings of the 31st Annual Conference on Advances in Cryptology, CRYPTO '11*, pp. 151–165. Springer-Verlag, Berlin. DOI: 10.1007/978-3-642-22792-9_9. 695

W. Diffie and M. E. Hellman. 1976. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6): 644–654. DOI: 10.1109/TIT.1976.1055638. 693, 701

Y. Z. Ding, D. Harnik, A. Rosen, and R. Shaltiel. 2007. Constant-round oblivious transfer in the bounded storage model. *Journal of Cryptology*, 20(2): 165–202. DOI: 10.1007/s00145-006-0438-1. 721

Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. 2008. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal of Computing*, 38(1): 97–139. DOI: 10.1137/060651380. 699, 700

S. Dziembowski and K. Pietrzak. 2008. Leakage-resilient cryptography. In *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pp. 293–302. IEEE Computer Society, Washington, DC. DOI: 10.1109/FOCS.2008.56. 695

B. Fuller, A. O'Neill, and L. Reyzin. 2015. A unified approach to deterministic encryption: New constructions and a connection to computational entropy. *Journal of Cryptology*, 28(3): 671–717. DOI: 10.1007/s00145-013-9174-5. 695

O. Goldreich. 2001. *Foundations of Cryptography: Volume 1, Basic Techniques*. Cambridge University Press, Cambridge. 723

O. Goldreich. 2008. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, Cambridge. 695

O. Goldreich. 2019. On the foundations of cryptography. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, Chapter 17. Association for Computing Machinery and Morgan & Claypool. This volume. 695

O. Goldreich and H. Krawczyk. 1992. Sparse pseudorandom distributions. *Random Structures & Algorithms*, 3(2): 163–174. DOI: 10.1002/rsa.3240030206. 704

O. Goldreich and L. A. Levin. 1989. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pp. 25–32. Seattle. DOI: 10.1145/73007.73010. 695, 706, 709

O. Goldreich and B. Meyer. 1998. Computational indistinguishability: Algorithms vs. circuits. *Theoretical Computer Science*, 191(1–2): 215–218. DOI: 10.1016/S0304-3975(97)00162-X. 705

O. Goldreich and S. Micali, 1984. Unpublished manuscript. See Goldreich [2001, Sec. 3.3.2]. 703

O. Goldreich, S. Goldwasser, and S. Micali. 1986. How to construct random functions. *Journal of the ACM*, 33(4): 792–807. DOI: 10.1145/6490.6503. 693

O. Goldreich, S. Micali, and A. Wigderson. 1991. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3): 690–728. DOI: 10.1145/116825.116852. 693

S. Goldwasser and S. Micali. 1984. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2): 270–299. DOI: 10.1016/0022-0000(84)90070-9. 693, 702

S. Goldwasser, S. Micali, and R. L. Rivest. 1988. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2): 281–308. Preliminary version in *FOCS '84*. DOI: 10.1137/0217017. 693

I. Haitner and S. P. Vadhan. 2017. The many entropies in one-way functions. In Y. Lindell, editor, *Tutorials on the Foundations of Cryptography—Dedicated to Oded Goldreich*, pp. 159–217. Springer. Also posted as *ECCC* TR17-084. DOI: 10.1007/978-3-319-57048-8_4. 695

I. Haitner, M. Nguyen, S. Ong, O. Reingold, and S. Vadhan. 2009a. Statistically hiding commitments and statistical zero-knowledge arguments from any one-way function. *SIAM Journal on Computing*, 39(3): 1153–1218. DOI: 10.1137/080725404. 693, 694, 696, 718

I. Haitner, O. Reingold, S. Vadhan, and H. Wee. 2009b. Inaccessible entropy. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pp. 611–620. ACM, New York. DOI: 10.1145/1536414.1536497. 694, 695, 717, 719, 720

I. Haitner, T. Holenstein, O. Reingold, S. Vadhan, and H. Wee. 2010. Universal one-way hash functions via inaccessible entropy. In *Proceedings of the 29th Annual International Conference on Theory and Applications of Cryptographic Techniques, EUROCRYPT '10*, pp. 616–637. Springer-Verlag, Berlin. DOI: 10.1007/978-3-642-13190-5_31. 694

I. Haitner, O. Reingold, and S. Vadhan. 2013. Efficiency improvements in constructing pseudorandom generators from one-way functions. *SIAM Journal on Computing*, 42(3): 1405–1430. DOI: 10.1137/100814421. 694, 695, 705, 710, 714, 717

I. Haitner, K. Nissim, E. Omri, R. Shaltiel, and J. Silbak. 2018. Computational two-party correlation: A dichotomy for key-agreement protocols. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science*, pp. 136–147. DOI: 10.1109/FOCS.2018.00022. 695

J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. 1999. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4): 1364–1396. DOI: 10.1137/S0097539793244708. 693, 694, 697, 700, 703, 704, 705, 706

T. Holenstein and R. Renner. 2011. On the randomness of independent experiments. *IEEE Transactions on Information Theory*, 57(4): 1865–1871. DOI: 10.1109/TIT.2011.2110230. 700, 701

C.-Y. Hsiao, C.-J. Lu, and L. Reyzin. 2007. Conditional computational entropy, or toward separating pseudoentropy from compressibility. In M. Naor, editor, *Advances in Cryptology—EUROCRYPT 2007*, pp. 169–186. Springer, Berlin. DOI: 10.1007/978-3-540-72540-4_10. 706, 707, 708

R. Impagliazzo. 1995. Hard-core distributions for somewhat hard problems. In *36th Annual Symposium on Foundations of Computer Science*, pp. 538–545. IEEE, Milwaukee. DOI: 10.1109/SFCS.1995.492584. 710

R. Impagliazzo and M. Luby. 1989. One-way functions are essential for complexity based cryptography. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pp. 230–235. IEEE Computer Society, Washington, DC. DOI: 10.1109/SFCS.1989.63483. 693

T. Itoh, Y. Ohta, and H. Shizuya. 1997. A language-dependent cryptographic primitive. *Journal of Cryptology*, 10(1): 37–49. DOI: 10.1007/s001459900018. 696

A. R. Klivans and R. A. Servedio. 2003. Boosting and hard-core set construction. *Machine Learning*, 51(3): 217–238. DOI: 10.1023/A:1022949332276. 710

D. Micciancio and S. Vadhan. 2003. Statistical zero-knowledge proofs with efficient provers: Lattice problems and more. In D. Boneh, editor, *Advances in Cryptology—CRYPTO '03*, vol. 2729 of Lecture Notes in Computer Science, pp. 282–298. Springer-Verlag. DOI: 10.1007/978-3-540-45146-4_17. 696

I. Mironov, O. Pandey, O. Reingold, and S. Vadhan. 2009. Computational differential privacy. In S. Halevi, editor, *Advances in Cryptology—CRYPTO '09*, vol. 5677, Lecture Notes in Computer Science, pp. 126–142. Springer-Verlag. DOI: 10.1007/978-3-642-03356-8_8. 695

M. Naor. 1991. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2): 151–158. DOI: 10.1007/BF00196774. 693, 718, 719

M. Naor and M. Yung. 1989. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pp. 33–43. ACM, New York. DOI: 10.1145/73007.73011. 693, 694, 722

M. Naor, R. Ostrovsky, R. Venkatesan, and M. Yung. 1998. Perfect zero-knowledge arguments for NP using any one-way permutation. *Journal of Cryptology*, 11(2): 87–108. DOI: 10.1007/s001459900037. 721

M. Nguyen and S. Vadhan. 2006. Zero knowledge with efficient provers. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pp. 287–295. DOI: 10.1145/1132516.1132559. 696

N. Nisan and D. Zuckerman. 1996. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1): 43–52. DOI: 10.1006/jcss.1996.0004. 697

S. J. Ong and S. Vadhan. 2007. Zero knowledge and soundness are symmetric. In M. Naor, editor, *Advances in Cryptology—EUROCRYPT '07*, vol. 4515, Lecture Notes in Computer Science, pp. 187–209. Springer-Verlag. DOI: 10.1007/978-3-540-72540-4_11.pdf. 696

J. Radhakrishnan and A. Ta-Shma. 2000. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM Journal on Discrete Mathematics*, 13(1): 2–24, (electronic). DOI: 10.1137/S0895480197329508. 698

O. Reingold, L. Trevisan, and S. Vadhan. 2004. Notions of reducibility between cryptographic primitives. In M. Naor, editor, *Proceedings of the First Theory of Cryptography*

Conference, TCC '04, vol. 2951, Lecture Notes in Computer Science, pp. 1–20. Springer-Verlag. DOI: 10.1007/978-3-540-24638-1_1. 703

R. Renner and S. Wolf. 2005. Simple and tight bounds for information reconciliation and privacy amplification. In *Proceedings of the 11th International Conference on Theory and Application of Cryptology and Information Security, ASIACRYPT '05*, pp. 199–216. Springer-Verlag. DOI: 10.1007/11593447_11. 698

J. Rompel. 1990. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pp. 387–394. ACM, New York. DOI: 10.1145/100216.100269. 693, 694, 722

M. Skórski, A. Golovnev, and K. Pietrzak. 2015. Condensed unpredictability. In M. M. Halldórsson, K. Iwama, N. Kobayashi, and B. Speckmann, editors, *Automata, Languages, and Programming*, pp. 1046–1057. Springer. DOI: 10.1007/978-3-662-47672-7_85. 710

M. Sudan, L. Trevisan, and S. Vadhan. 2001. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62: 236–266. 710

A. Ta-Shma and D. Zuckerman. 2004. Extractor codes. *IEEE Transactions on Information Theory*, 50(12): 3015–3025. DOI: 10.1109/TIT.2004.838377. 708

L. Trevisan. 2001. Extractors and pseudorandom generators. *Journal of the ACM*, 48(4): 860–879 (electronic). DOI: 10.1145/502090.502099. 708

S. Vadhan and C. J. Zheng. 2012. Characterizing pseudoentropy and simplifying pseudorandom generator constructions. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing*, pp. 817–836. ACM, New York. DOI: 10.1145/2213977.2214051. 694, 695, 710, 712, 713, 715, 718

S. P. Vadhan. 1999. *A Study of Statistical Zero-Knowledge Proofs*. Ph.D. thesis, Massachusetts Institute of Technology. 696

S. P. Vadhan. 2012. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1–3): 1–336. DOI: 10.1561/0400000010. 695, 698, 699, 708

A. C. Yao. 1982. Theory and application of trapdoor functions. In *23rd Annual Symposium on Foundations of Computer Science*, pp. 80–91. DOI: 10.1109/SFCS.1982.45. 693, 695, 702, 706

C. J. Zheng. 2014. *A Uniform Min-Max Theorem and Characterizations of Computational Randomness*. Ph.D. thesis, Harvard University. http://nrs.harvard.edu/urn-3:HUL.InstRepos:11745716. 710