Aalto University
School of Science
Master's Programme in Mathematics and Operations research

Miikka Tiainen

Computational entropy from distributional hardness

Master's Thesis
Espoo, November 9, 2021

Supervisor: Professor Christopher Brzuska
Advisor: Professor Christopher Bruzska

Aalto University
School of Science
Master's Programme in Mathematics and Operations research

ABSTRACT OF
MASTER'S THESIS

Author: Miikka Tiainen

Title:
Computational entropy from distributional hardness

Date: November 9, 2021 **Pages:** 69

Major: Mathematics **Code:** SCI3054

Supervisor: Professor Christopher Brzuska

Advisor: Professor Christopher Bruzska

A central problem in cryptography is the construction of basic *primitives*, or low-level algorithms, from computational complexity-based assumptions. One way of viewing the hardness of a problem from the view of a computationally bounded adversary is via the notion of *entropy*. Much like the toss of a normal coin is considered random due to limitations of human observers, the real entropy, or uncertainty, of a system can be much higher or lower than the entropy that is "observable" by an efficient adversary.

In this thesis we establish results obtaining this type of computational entropy from *distributionally hard* primitives. This notion of distributional hardness captures that it is hard for an adversary to output a uniform pre-image of a randomly sampled image value. We use this computational entropy from distributional hardness to expand on existing results constructing pseudorandom generators from *next-block pseudoentropy*, and statistically hiding commitment schemes from *accessible entropy*. Although the existence of such constructions were implicit in existing literature, we establish much more efficient constructions with tighter bounds on the computational entropy than has previously been considered. Furthermore, the current known optimal construction of pseudorandom generators in terms of input (or *seed*) length appears to hold with equivalent parameters for the much more general notion of distributional hardness, establishing that the much more general notion of distributional hardness may in itself yield conceptually interesting constructions.

We improve on existing results using known known information theoretic inequalities. Most centrally we use an inequality due to Bretagnolle and Huber relating the statistical distance and relative entropy of distributions in a much tighter way in the context of highly disjoint distributions than the famous Pinsker bound.

Keywords: One-way functions, Hash-functions, Statistical distance, Computational entropy, Relative entropy, Tightness

Language: English

Aalto-universitetet
Högskolan för teknikvetenskaper
Magisterprogrammet i matematik

SAMMANDRAG AV
DIPLOMARBETET

Utfört av: Miikka Tiainen

Arbetets namn:
Computational entropy from distributional hardness

Datum:	Den 9 november 2021	**Sidantal:**	69
Huvudämne:	Matematik	**Kod:**	SCI3054
Övervakare:	Professor Christopher Brzuska		
Handledare:	Professor Christopher Brzuska		

Ett centralt problem inom kryptografi är konstruktionen av kryptografiska *primitiv*, eller enkla algorimtmer, från kompleksitetsteoretiska antaganden. Ett sätt att uppfatta svårhetsgraden av ett problem för en motståndare med begränsade beräkningsresurser är genom begreppet *entropi*. Liksom deterministiska (och därmed förutsägbara) myntkast behandlas som slumpmässiga för mänskilga observatörer, kan den äkta entropin, eller osäkerheten, av ett kryptosystem vara mycket lägre eller högre än vad en begränsad motståndare klarar av att observera.

I detta diplomarbete härleder vi resultat som etalberar denna typs beräknelig entropi från *distributionellt svåra* primitiv. Distributionell svårhet innebär att det är svårt för en motståndare att hitta jämndistribuerade urbilder för slumpmässiga bildelement. Detta inkluderar även funktioner som alltid har en enkel urbild. Via beräknelig entropi från distributionellt svåra problem bygger vi på existerande resultat inom litteraturen och påvisar konstruktioner av pseudoslumpgeneratorer från *pseudoentropi*, och statistiskt gömmande lojalitetssystem från *tillgänglig entropi*. Dessa konstruktioner är implicita i den existerande litteraturen, men våra resultat etablerar mycket stramare konstruktioner via mera optimala olikheter för beräknelig entropi från distributionell svårhet. Intressant nog, verkar den just nu mest optimala konstruktionen av pseudoslumpgeneratorer från envägsfunktioner hålla med ekvivalenta parametrar för den mycket mer generalla klassen av distributionellt hårda funktioner, alltså kan distributionellt svåra primitiv redan i sig möjligen ge interessanta konstruktioner.

Vi bygger på existerande resultat via kända informationsteoretiska olikheter, mest centralt en olikhet av Bretagnolle och Huber som etablerar ett samband mellan statistisk distans of relativ entropi.

Nyckelord:	Envägsfunktioner, hashfunktioner, statistisk distans, beräknelig entropi, relativ entropi, stramhet
Språk:	Engelska

Contents

Chapter 1

Introduction

The strongest notion of security achievable by a cryptographic system or protocol is that of *information theoretic* security, where all adversaries are assumed to be computationally unbounded. This however places inherent limitations on what type of cryptography is possible. For example, private key encryption is impossible with information theoretic security, and symmetric encryption always requires keys as long as the message. Much of modern cryptographic security is instead based on complexity theory based assumptions of hardness, where adversaries are assumed to have bounded computational resources. The simplest, most foundational property which separates *computational* security from information theoretic security appears to be that of *one-wayness*. Impagliazzo and Luby showed that one-way functions are *necessary* for much of complexity based cryptography [22]. Namely, one-way functions are easy to compute but infeasible to invert for any (polynomially) bounded adversary, whereas one-wayness is plainly impossible against an unbounded adversary. On the other hand the existence of one-way functions has been shown to imply the existence, among other things, pseudorandom generators [21], pseudorandom functions [16], signature schemes [26], coin flipping protocols [22] and further useful primitives. These constructions are also necessary from a foundational perspective, as one-way functions by themselves provide little security guarantees. They may leak large (indeed any constant fraction) parts of their input, contain trivial constant substrings in their output or ignore large parts of the input, and nevertheless be infeasible to invert. Thus there is a need to construct primitives with more concrete/specific security guarantees to obtain building blocks for practically useful cryptography.

In the past decade, a line of research ([18, 19, 17, 37, 1]) has culminated in constructions of primitives from one-way functions via means of *computational entropy*. Intuitively, from the computational hardness of a one-way

function it is possible to obtain uncertainty, or *entropy*, from the perspective of computationally bounded adversaries. For example, for a bijective function f that is infeasible to invert, knowing any image $f(x)$ fixes the pre-image x information theoretically, but by definition no computationally bounded adversary can return this value with noticeable probability. That is to say, the entropy of x given $f(x)$ is in reality zero, but x *appears* uncertain given $f(x)$ to any adversary. Computational entropy can then be structured and transformed to obtain new random variables (i.e. cryptographic primitives) with strong properties, such as *pseudorandomess*, which we want for pseudorandom generators, and *unforgeability*, which we want for statistically hiding commitments.

The focus of this thesis will be the study of *(next-block) pseudoentropy* and its connection to pseudorandom generators (PRGs), as well as *accessible entropy* and its connection to the construction of statistically hiding commitments. The notions of both accessible entropy and pseudoentropy also seem to have a fundamental connection to that of *relative entropy* ([1],) an f-divergence (cf. [33]) used in information theory to measure *closeness* of distributions. Namely, the underlying uniform distribution given as input to any standard one-way function can be shown to have high relative entropy with respect to the output distribution of a bounded adversary trying to find a valid pre-image behind the one-way function, i.e. the divergence between the distributions $(x, f(x))$ and $(\mathcal{A}(f(x)), f(x))$ is high when f is sufficiently hard to invert. Using amplification techniques, bounds on the relative entropy can then further be used to obtain bounds on both accessible entropy and pseudoentropy. In other words, the *closeness* of the uniform and adversarial distributions allows us to obtain computational entropy from hardness assumptions.

Our central question will be whether the connection between computational hardness and relative entropy also to generalizes to *distributional* hardness where we quantify the advantage of an adversary in terms of the *statistical distance*. Phrased differently, we demand that the adversary should be able to sample *unifromly* from the set of valid solutions. For example, can a function be *always* easy to invert, and still yield useful entropy bounds as long as the function has sufficiently strong *distributional* hardness guarantees? Intuitively, if a function always has one easy pre-image, but say 9 infeasible ones, an adversary should still fail to sample roughly $\frac{9}{10}$ of the pre-image distribution. In other words, even though the support of the adversary is fully contained in the input distribution, the support of the adversary is skewed. We will show that the distributional *hardness* yields very strong bounds on relative entropy, implying a construction of PRGs from *distributional one-way functions* that is *equivalent* to the best known from standard

one-way functions in terms of input (or *seed*) length, as well as an improvement on the accessible entropy obtainable from *distributionally collision resistant hash functions*. Additionally we present a central inequality due to Bretagnolle and Huber [8], which has yet to attract much use in cryptography, that is our main inequality for relating statistical distance to relative entropy. We address the full scope, specific results and research questions of the thesis more thoroughly in Section 1.2.

We proceed by giving brief background to the notions of *pseudoentropy*, *accessible entropy* and *distributional one-way functions* as well as *distributionally collision resistant hash functions*. These four notions are the foundational definitions for the thesis.

1.1 Background

1.1.1 Pseudoentropy

We first recall the notions of *Shannon entropy* and *relative entropy* that are necessary for formally defining both next-block pseudoentropy and accessible entropy. Throughout this thesis we refer to the entropy of *random variables* when meaning the entropy of their induced probability distributions. This terminology is standard is cryptographic literature.

Definition 1.1.1 (Shannon Entropy)**.** *The Shannon entropy* $\mathrm{H}(X)$ *of a random variable* X *is defined as*

$$\mathrm{H}(X) = \mathbb{E}_X\left[\log \frac{1}{\Pr[X = x]}\right] = \sum_{x \in \mathrm{Supp}(X)} \Pr[X = x] \cdot \log \frac{1}{\Pr[X = x]}.$$

Intuitively, Shannon entropy can be understood as the average uncertainty of an outcome. We also require the notion of *conditional entropy*:

Definition 1.1.2 (Conditional entropy)**.** *The conditional entropy of* X *given* Y *is defined as*

$$\mathrm{H}(X \mid Y) = \mathbb{E}_y[\mathrm{H}(X|Y = y)].$$

Conditional entropy can also be used to characterize the entropy of a joint random variable, often referred to as the *chain rule*:

Lemma 1.1.1 (Chain rule of Shannon entropy)**.** *For a joint random variable* $X = (X_1, ..., X_m)$ *it holds that the entropy of* X *is the sum of the conditional entropies:*

$$H(X) = \sum_{i=1}^{n} \mathrm{H}(X_i \mid X_{<i}),$$

this is known as the chain rule of conditional entropy.

Compared to Shannon entropy, which is a functional of the underlying distribution, the *relative entropy* of random variables instead formalizes *closeness* of two random variables:

Definition 1.1.3 (Relative entropy/KL divergence)**.** *For random variables X and Y on a discrete and finite sample space we define the relative entropy (or Kullback-Leibler divergence) from X to Y as*

$$\mathsf{KL}(X\|Y) = \sum_{x \in \mathrm{Supp}(X)} \Pr[X = x] \cdot \log \frac{\Pr[X = x]}{\Pr[Y = x]},$$

where by convention $\mathsf{KL}(X\|Y) = \infty$ if $\Pr[Y = x]$ is zero when $\Pr[X = x]$ is nonzero.

Note that the KL divergence is not symmetric, and is as such not a proper metric, as it ignores probabilities of Y on events when X has zero probability. We now proceed by presenting the basic definitions of pseudoentropy and accessible entropy.

1.1.1.1 HILL pseudoentropy

One-way functions (OWFs) are efficiently computable functions that are computationally infeasible to invert (see Def. 2.1.1). A seminal result due to due to Håstad, Impagliazzo, Luby and Levin [21] establishes that one-way functions imply pseudorandom generators (PRGs). In essence, they show that from computational hardness it is possible to obtain "random-looking" strings. Formally, PRGs are defined as efficient length extending functions, whose output is indistinguishable from a random string (see Def. 2.1.2). The main tool introduced by HILL is the notion of (HILL) pseudoentropy, a precursor of the notion of *next-block pseudoentropy* that we will use in this thesis. The central idea is that when interacting with a computationally bounded adversary a random variable may *appear* to have more entropy than it does in reality. This idea is formally defined via *computational indistinguishability*.

Definition 1.1.4 (Computational Indistinguishability)**.** *We say random variables X and Y are computationally indistinguishable if for all probabilistic polynomial time distinguishers $\mathcal{D}$ it holds that*

$$\mathsf{Adv}_{\mathcal{D}}(n) := |\Pr[\mathcal{D}(X, 1^n) = 1] - \Pr[\mathcal{D}(Y, 1^n) = 1)]| = \mathsf{negl}(n).$$

We also write $X \overset{c}{\approx} Y$ to denote computational indistinguishability of the random variables X and Y. The function $\mathsf{negl}(n)$ denotes a negligible function, i.e. a function that is defined as being asymptotically smaller than the inverse of any positive polynomial in n. Note that X and Y can be taken as single random variables, but we will mainly consider them as *sequences* of random variables indexed by some function of n.

Definition 1.1.5 (HILL Pseudoentropy)**.** *A random variable X has pseudoentropy k if there exists some random variable Y such that*

$$X \overset{c}{\approx} Y, \text{ and } \mathrm{H}(Y) \geq k.$$

In other words, since no efficient distinguisher $\mathcal{D}$ can distinguish X from Y, and Y has entropy k, then X will also appear to have entropy k. Particularly interesting is the case when the pseudoentropy threshold k is larger than the real entropy of X. This difference $k - \mathrm{H}(X)$ is called the *gap* in pseudoentropy.

1.1.1.2 Pseudoentropy and next-block pseudoentropy

The notion of pseudoentropy was further generalized by Haitner, Reingold and Vadhan [18] to that of next-block pseudoentropy (nbpe), where we divide a random variable into blocks $X = (X_1, ..., X_m)$, and demand that conditioned on all previous blocks up to some index i, one should not be able to distinguish the i-th block from a high entropy random variable.

Definition 1.1.6 (Next-block pseudoentropy (nbpe))**.** *An m-block random variable $X = (X_1, ..., X_m)$ has next-block pseudoentropy (at least) k if it holds that for $i \in [m]$ each X_i is computationally indistinguishable from some Y_i when conditioned on previous blocks $X_1, ..., X_{i-1}$, where $Y = (Y_1, ..., Y_m)$ is jointly distributed with X and*

$$\sum_{i=1}^{m} \mathrm{H}(Y_i | X_1, ..., X_{i-1}) \geq k.$$

We can analogously extend the definition to a conditional random variable $X = A|B$ and say that A has conditional next-block pseudoentropy at least k given B. In the case of one bit blocks we refer to next *bit* pseudoentropy. We will be particularly interested in the next-block pseudoentropy of the induced output distribution $G(x)$ of functions G with m-block outputs on uniformly random inputs x, formalized as *next-block pseudoentropy generators*. Note that as evaluating a deterministic function on a uniformly random input is

itself a random variable, the notion of dividing a random variable X into m blocks $X = (X_1, ..., X_m)$ is directly applicable for the output values of the generator.

Definition 1.1.7 (Next-block pseudoentropy generator)**.** *A polynomial time computable function $G : \{0,1\}^n \to (\{0,1\}^{l(n)})^{m(n)}$ with $m(n)$ output blocks of (possibly varying) length $l(n)$ is called a next-block pseudoentropy generator if it has next-block pseudoentropy at least $k = \mathrm{H}(G(x)) + \delta$ for uniformly random bitstrings x and some noticeable δ. The function δ is called the pseudoentropy gap of the generator.*

A pseudorandom generator from n bits to m bits for example has real entropy at most n (based on the input being uniformly random bitstrings of length n), but is by definition computationally indistinguishable from a uniform bitstring of length $m > n$, meaning it can be shown to have next-block pseudoentropy m.

Pseudoentropy was shown by Vadhan and Zheng to be *equivalent* to the notion of *relative pseudoentropy*, in that X has relative pseudoentropy Δ given Y if and only if X has an pseudoentropy $H(X|Y) + \Delta$ given Y. We define relative pseudoentropy as follows:

Definition 1.1.8 (relative pseudoentropy [37] definition 3.4, [1] Definition 3.5)**.** *For a jointly distributed random variable (Y, X) with security parameter n, we say that X has $\Delta(n) := \Delta$ relative pseudoentropy given Y if for all PPT adversaries $\mathcal{A}$ it holds that*

$$\mathsf{KL}((Y, X) \| (Y, \mathcal{A}(Y))) > \Delta.$$

In other words X has relative pseudoentropy given Y if every efficient adversary fails to approximate the conditional distribution of X given Y up to some lower bound Δ when measured in relative entropy. This notion was introduced as *KL-hardness for sampling* [37], but later papers following this line of results have adopted the term *relative pseudoentropy*, which we also use in this thesis (e.g. [1]).

1.1.2 Accessible entropy

Whereas the pseudoentropy of an m-block generator G is defined as a distinguishing problem between the output and some high entropy variable, the accessible entropy of G is instead defined via adversary trying generate (or *emulate*) the distribution induced by G. The accessible entropy of a generator G is the maximal entropy of any adversarial emulator $\widetilde{G}$ trying to emulate

G, whose output is always in the support of G (defined as "*G-consistent*" in the original work by Haitner, Reingold, Vadhan and Wee [19]). We note that in the context of accessible entropy, when writing m-block generator we simply mean a deterministic function taking as input a public parameter p and some seed s that outputs m blocks of arbitrary length. In turn, in order to output the i-th block $\widetilde{G}[i]$, the emulator (or adversarial generator) $\widetilde{G}$ is further allowed to sample new randomness for the seed, conditioned on its previous output blocks (i.e. it is allowed to "cheat" in an online fashion). The i-th output block only depends on the previous internal coin tosses.

The gap between the real entropy of G (i.e. the Shannon entropy of the induced output distribution) and its accessible entropy is called *inaccessible entropy*, due to the fact that no poly time adversary trying to pass itself off as G can successfully "access" the full entropy of the distribution. Any generator G that has noticeable inaccessible entropy is called an *inaccessible entropy generator*. Accessible entropy of $\widetilde{G}$ is meaningful when it is noticeably *smaller* than the real entropy of G. Due to computational hardness, we may construct generators G with a gap in real and accessible entropy.

Definition 1.1.9 (Accessible entropy of generator G)**.** *Let $G : \{0,1\}^n \to (\{0,1\}^*)^m$ be a function with m output blocks and denote the the output of $G(x)$ on a uniformly random x by $Y = (Y_1, ..., Y_m)$.*

Let $\widetilde{G}$ be a poly time probabilistic algorithm that on public parameter P and some internal coin tosses $(R_1, ..., R_2)$ outputs $\widetilde{Y} = (\widetilde{Y_1}, ..., \widetilde{Y_m})$ such that the support of $\widetilde{G}$ is contained in that of G . The accessible entropy of $\widetilde{G}$ is defined as

$$\sum_{i=1}^{m} \mathrm{H}\Big(\widetilde{G}[i] \;\Big|\; P, R_{<i}\Big) = \sum_{i=1}^{m} \mathrm{H}\Big(\widetilde{Y_i} | P, R_1, ..., R_{i-1}\Big)$$

If the accessible entropy of all poly time G-consistent emulators is bounded from above by some $k_{acc}(n) > 0$ we say that G has accessible entropy $k_{acc}(n)$.

The public parameter P could be, for example, a description for a hash function used in the generator.

Note that the requirement of the emulator being consistent is vital, as otherwise an adversary that outputs arbitrarily long strings could be used to obtain arbitrarily large accessible entropy, despite it producing outputs that are unrelated to the underlying generator.

A simple illustrative example that is given by Haitner, Reingold, Vadhan and Wee [19] is that of the two block generator $x \mapsto (h(x), x)$, where h is a *target collision resistant hash function* (where given a value $h(x)$ it is computationally infeasible to sample two values that map to $h(x)$) on a

uniformly random bitstring x of length n. The real entropy of x given $h(x)$ is at least $n - |h(x)|$. Regardless, due to the collision resistance no poly-time algorithm (outputting strings in the support of $h(x), x$) can have more that a negligible amount of entropy for the second block given the first. This is due to that the first block effectively *determines* the second, as finding more than one valid pre-image of $h(x)$ under h is computationally infeasible. Thus even though the real entropy is at least $n - |h(x)|$, no efficient algorithm can *access* this entropy.

1.1.3 Distributional one-way functions

A relaxation of one-way functions as originally defined by Impagliazzo and Luby [22] is the notion of a *distributional* one-way function where the advantage of the adversary is measured as the statistical distance between the joint distributions $(f(x), x)$ and $(f(x), \mathcal{A}(f(x)))$, for a uniformly random x. For two discrete probability distributions p and q, their statistical distance (also called total variation distance) is defined as

$$\mathsf{SD}(p, q) = \frac{1}{2} \sum_{x \in \Omega} |p(x) - q(x)| = 1 - \sum_{x \in \Omega} \min(p(x), q(x)).$$

An ideal adversary against a distributional one-way function returns uniform pre-images for every image $f(x)$, and has statistical distance equal to zero. Some global lower bound on the statistical distance on the other hand implies that there is a some fraction of correct pre-images that no efficient adversary can put significant probability mass on given the image.

Definition 1.1.10 (Distributional one-way function)**.** *We say that an efficiently computable function f is $\delta(n)$-distributionally one-way if for every probabilistic polynomial time adversary $\mathcal{A}$ it holds for some noticeable $\delta := \delta(n)$ and large enough n we have that*

$$\mathsf{SD}_{x \xleftarrow{\$} \{0,1\}^n}((f(x), x), (f(x), \mathcal{A}_f)) > \delta,$$

where we denote $\mathcal{A}_f = \mathcal{A}(f(x))$ (note that the randomness of the adversary also affects the statistical distance).

If f is $1 - \mathsf{negl}(n)$-distributionally one-way we say it is a *strong distributional one-way function*.

As discussed earlier, a distributional notion of hardness allows a function to have a lot of hardness, even if we can always invert it, as long as a significant portion of the input distribution remains hard for any efficient adversary to return.

1.1.3.1 Distributionally one-way problems

It is immediate that any (standard) one-way function has $1 - \mathsf{negl}(n)$ statistical distance in the above sense, as all but a negligible amount of the adversarial mass has to be on invalid pre-image/image pairs [1], implying high statistical distance from the uniform distribution over valid pre-images. Below, we define a folklore example of a strong distributional OWF. Namely, we define a function $f' : \{0,1\}^n \to \{0,1\}^{n/2}$ such that

$$f'(x_l||x_r) = \begin{cases} x_r, & \text{if } x_l = 0...0 \\ f(x_r), & \text{for some strong one-way function } f \text{ otherwise} \end{cases} \tag{1.1}$$

The function f' is clearly $1 - \mathsf{negl}(n)$ distributionally one-way, as the easy pre-images make up an exponentially small fraction of the input distribution, while f is also invertible for each image by prepending zeros to the image. On the other hand the example is not very satisfactory − it assumes a strong one-way function and introduces easy pre-images, rather than being a "natural" problem that admits to some easy solutions, but does not allow an adversary to sample uniform pre-images. Thus, we now discuss two scenarios where distributional one-way functions come up naturally, and then review distributional OWF in literature.

Computationally indistinguishable but statistically distinguishable distributions (as shown in eg. [27, 5]). In this case we define a function using a pair of statistically close, but computationally far apart distributions D_0, D_1 that are efficiently samplable. On input (b, r) the function outputs the distribution b sampled using randomness r, i.e. $D_b(r)$. It is possible to show that if this is not a distributional one-way function then an assumed adversary can be used to efficiently distinguish between the distributions. Namely, if given a random distribution an adversary finds close to uniform pre-images it is possible to obtain a good predictor for the distributions D_0, D_1 by sampling enough pre-images and taking the majority [2].

Unsupervised learning (cf. Boaz Barak's survey on one-way functions [3]). Consider a one-way function based on the hardness of learning parity with noise (LPN) as defined in Pietrzak [30]: define a public expanding matrix A mapping short vectors to long vectors, and let $f(s, e) = (A, As + e)$,

[1] Therefore, if a strong distributional one-way function is injective then it is a standard one-way function.

[2] In simple cases outputting the pre-image bit usffices, but as shown by Berman, Degwekar, Rothblum and Vasudevan in some regimes it is necessary to take many samples to obtain a good estimate [5]

where + denotes bitwise XOR. If we place no strict upper bound on the Hamming weight of the noise e, we may invert any valid image (A, y) using zero secret $s = (0, ...0)$ and noise vector $e = y$. This nonetheless doesn't affect the conceptual hardness of the LPN problem. Similarly to the function f' in the pathological example in eq. (1.1), even though the function f' is invertible with probability 1, all the mass is on an exponentially small fraction of the pre-images. If we suppose towards contradiction that f is not distributionally one-way, then this immediately implies that there exists an adversary that places some non-negligible mass on the "hard" LPN pre-images with a nontrivial, nonzero secret s.

1.1.3.2 Distributional one-way functions in the literature

Definition. The original definition of distributional one-way functions is due to Impagliazzo and Luby [23], who also show the equivalence of one-way functions and distributional OWFs. The central use in the original paper is showing that a wide variety of cryptographic primitives directly imply distributional OWFs (and thus OWFs are *necessary* for complexity based cryptography). Similar use of distributional OWFs, namely showing that some hardness assumption implies or is equivalent to OWFs, remains a (or even *the*) central use of distributional one-wayness in current literature. Several constructions of primitives also make use of distributional OWFs directly instead of standard OWFs. Establishing that a primitive can either be directly used as a distributional OWF, or that one can transform it into a distributional OWF implicitly also means that the primitive is existentially equivalent to standard OWFs. Beimel, Ishai, Kushilevitz and Malkin [4] show that the existence of a *Single-Server Private Information Retrieval protocol* implies OWFs in this way. Similarly a line of results regarding coin-flipping protocols (see cf. [6]), shows that secure coin-flipping protocols must be secure against uniform inversion. Appelbaum, Ishai and Kushilevitz [2] show that a robust OWF (where a function remains hard to invert even if for a random subset of output bits the effective input bits are revealed) can be transformed into a distributional OWF by semi-randomized private encoding, which is then further fashioned into a OWF by the results of Impagliazzo and Luby. Pietrzak and Sjödin [31] use the notion of a distributional one-way function sampling inputs to a *weak* PRF that does not have secret coins [3] to establish that a secret-coin (used to sample the inputs) weak PRF that is not a weak PRF with public coins implies the existence of public key encryption.

[3] A PRF is *weak* when indistinguishable from a random function on uniformly random inputs, as opposed to arbitrary adversarially chosen inputs

This in turn means, that if OWFs exist, but public key encryption does not, then the notions of public- and secret- coin weak PRFs are equal.

Statistical zero knowledge. In the area of *statistical zero-knowledge* languages (SZK) [4], one way to establish relations between SZK to other complexity classes is via assuming a distributional inverter (which is then equivalent to one-way functions not existing). This ties the existence of zero knowledge proofs for a given language to the existence of one-way functions. Ostrovsky [28] showed that a hard on average SZK language L implies a distributional one-way function in this fashion. In follow-up work, Ostrovsky and Widgerson show that if a very weak form of OWFs that they call *auxiliary input* OWFs do not exist, then only languages in BPP have zero-knowledge proofs. [29].

Inversion oracle. In another language context, Bogdanov and Brzuska in turn assume a distributional inverter in their proof of the impossibility of basing size-verifiable one-way functions on NP-hardness. Specifically their reduction makes use of an oracle U, that on input y returns uniformly from the set $f^{-1}(y)$, which in turn is used to argue that if there exists a black-box reduction which turns this inverter against a OWF to a decision algorithm for some language L, then L is contained in AM $\cap$ coAM. By widely believed complexity assumptions, this means L can not be NP-complete. Another use of dOWFs in the context of languages is by Pass and Venkitasubramaniam, who show that it is just as hard to find witnesses for efficiently sampled true statements, as opposed to sampling statements not even guaranteed to have a valid witness. They achieve this via connecting the existence of OWFs (and thus dOWFs) and k-round interactive *puzzles* (which are further equivalent to hard-on-the-average languages in NP).

Statistical versus computational distinguishability. Distributional one-way functions also appear in literature whenever there appears some gap between statistical and the end of Section 1.1.3.1. Goldreich originally established that one-way functions exist if and only if there exist families of distributions that are statistically far (i.e. the statistical distance is bounded from below by an inverse polynomial) but computationally indistinguishable in this fashion [15]. Similarly Naor and Rothblum provide a tighter result for the same claim [27]. Berman, Degwekar, Rothblum and Vasudevan use a similar proof to establish that the average case hardness of the so-called *Statistical Distance Problem* (SDP), where one is asked to distinguish if a pair of distributions D_0, D_1 are α-far or β-close, directly implies

[4] A language is said to be in SZK if there exists a pair of algorithms called prover and verifier, where the prover reveals no information other than the fact that the proof is correct, such that the probability distribution of the prover-verifier interaction is simulatable by unbounded adversaries.

distributional one-way functions for certain pairs (α, β). This is again done (essentially) via choosing a random bit b, and sampling from the distribution D_b. Any distributional adversary can then be used to solve the SDP problem. In his PhD thesis Vadhan showed that SDP is a complete problem for SZK.

Unsupervised learning. As mentioned previously, distributional-one way functions can also be connected to learning problems (cf. [3]). Namely, Xiao shows that "Learning to create is as hard as learning to appreciate" [40]. *Learning to create* or LTC is referred to as such due to the fact that given access to samples from a distribution D, an algorithm is is supposed to construct a circuit mimicking D as closely as possible. If LTC is solvable for circuits of size n^2, then no family of circuits is distributionally one-way. Thus if one-way circuits exist, then LTC is hard. Furthermore, one can transform a Probably Approximately Correct (PAC) learning problem (*learning to appreciate*), where given samples of a function an algorithm should be able to predict the value of unseen values, into an instance of the "circuit agnostic learning problem", which can be solved via a distributional inverter. Thus LTC is as hard as PAC learning. This connection is also pointed out in the PhD thesis of Xiao [39], which includes the aforementioned paper.

Computational entropy. Distributional one-way functions have also been discussed in the context of computational entropy. Namely, Vadhan and Zheng note that the notion of KL-hardness for sampling is similar to that of dOWFs [37], tying dOWFs to their construction of PRGs implicitly. Distributional one-way functions are also used to argue that the mere existence of an inaccessible entropy generator implies OWFs (again due to a distributional inverter breaking the accessible entropy bounds) [19]. Direct constructions and connections between the entropy notions and distributional one-way functions are not noted however.

1.1.4 Distributionally collision resistant hash-functions

Recently, the generalization of *distributional collision resistance* has also attracted some attention [12, 25, 7]. The notion of distributional collision resistance was originally defined by Dubrov and Ishai [12]. They relax the notion of collision resistance, the fact that it should be hard to sample collisions under a hash function h, to that it should be hard to sample *uniform* collisions. Dubrov and Ishai show that if it is not possible to efficiently sample any efficiently samplable distribution with as much randomness as the output, then given a one-way permutation, one can construct a distributionally collision resistant hash functions (dCRHF). Recently, constructions of dCHRFs have been obtained from both multi collision resistance and SZK [25]. Bitansky, Haitner, Komargodski and Yogev show how construct a statistically hiding

commitment scheme from a dCRHF[7]. For a dCRHF we measure the success as the statistical distance between the adversarial distribution and *ideal collision distribution.*

The distribution Col is a distribution defined by a hash family $\mathcal{H}$ such that $\mathsf{Col}(h, 1^n)$ outputs a pair (x_1, x_2), where x_1 is sampled uniformly at random from bitstrings of length 1^n, and x_2 is a uniformly random element of the pre-image set $h^{-1}(h(x))$.

Definition 1.1.11 (Ideal collision distribution)**.** *Let $\mathcal{H}$ be a family of hash functions. The ideal collision distribution with respect to $h \leftarrow\!\!\$\, \mathcal{H}$ is defined as $\mathsf{Col}(h) = \{(x_1, x_2) : x_1 \leftarrow\!\!\$\, \{0,1\}^n,\ x_2 \leftarrow\!\!\$\, h^{-1}(h(x))\}$.*

The definition is originally due to Simon [34], but was used specifically in the context of dCRHFs in BHKY. We note that by Simon's separation of collision resistant hash functions and OWFs also holds for dCRHFs (as collision resistance implies distributional collision resistance). As such, dCRHFs can not be constructed from OWFs in a black-box way. Note that Col may not be (and indeed in interesting cases is not) efficiently constructible.

Definition 1.1.12 (Distributionally collision resistant hash function family)**.** *A family of functions $\mathcal{H}$ is said to be a secure distributional collision resistant hash function family if there exists a polynomial p, such that for any PPT adversary $\mathcal{A}$ and $h \leftarrow\!\!\$\, \mathcal{H}$ it holds that*

$$\mathsf{SD}((h, \mathcal{A}(h, 1^n), (h, Col(h)) = \mathbb{E}_{h \leftarrow\!\!\$\, \mathcal{H}}[\mathsf{SD}(\mathcal{A}(h, 1^n), \mathsf{Col}(h)] > \frac{1}{p}$$

for large enough n. [5]

We say h is a distributionally collision resistant hash function (dCRHF) if it is sampled at random from such a family. Note that we may equivalently use a global lower bound written as $1 - \frac{1}{q(n)}$ for some polynomial $q(n)$, which allows us to analyze the setting where the function becomes harder as n tends to infinity.

BHKY show that although distributional collision resistance implies distributional one-wayness (and therefore OWFs), we can construct a constant round statistically hiding commitment (SHC) from dCRHF, whereas a known lower bound of $O(n/\log n)$ round commitment exists for OWF due to [LB SOURCE]. This is notably the first example of an application where dCRHF go beyond the power of one-way functions. The exact limitations of distributional collision resistance compared to stronger collision resistance assumptions such as multi-collision resistance or plain collision resistance are not yet

[5] To see that the equality holds conditioning the sum on the choice of h allows us to derive exactly the expectation on the left-hand side.

fully understood. Nonetheless, distributional collision resistance provides an additional way of defining hardness via statistical distance, aside from just distributional one-wayness.

1.1.5 Bretagnolle-Huber's inequality

Recall that we wish to establish bounds on the computational entropy obtainable from distributional hardness via relative entropy. These computational entropies will then further yield constructions of more practically useful primitives. When relating relative entropy and statistical distance, most commonly the famous Pinsker's inequality is used.

Lemma 1.1.2 (Pinsker's inequality)**.** *For two random variables X, Y it holds that*

$$\mathsf{SD}(X,Y) \leq \sqrt{\frac{\ln 2}{2}\mathsf{KL}(X\|Y)}$$

In the cases of our constructions however, the bound provided by Pinsker's inequality is problematic. It yields, at best a constant lower bound on the relative entropy. Especially when we have a high bound on the statistical distance (i.e. close to 1), Pinsker's inequality yields a very low bound for the relative entropy, and thus low bounds on computational entropy. This holds despite the fact that the distributions involved are almost disjoint (and thus should have *high* relative entropy). On the other hand, the following inequality due to Bretagnolle and Huber yields very useful bounds even in this high probability regime:

Lemma 1.1.3 ([8] Lemma 2.1, cf. [35], [33] Thm. 19)**.** *For any two probability distributions p, q defined on a common space it holds that*

$$\mathsf{SD}(p,q) \leq \sqrt{1 - 2^{\mathsf{KL}(p\|q)}}.$$

This inequality is also known as high probability Pinsker in some sources . In particular, if we can lower bound the statistical distance by $1 - 1/p(n)$ for some polynomial p, we obtain a lower bound on the relative entropy that grows with $p(n)$. This is *the* central inequality for our analysis in both improvements to existing constructions we aim to present in Chapter 3 and 4, as it allows for relative entropy bounds, yielding in turn computational entropy bounds, that grow as a function of n. As such we include it here and provide a separate proof in Appendix A.

1.2 Contribution

Our central question is whether current results obtaining computational entropy (pseudo- or inaccessible entropy) from distributionally hard primitives can be tightened or made explicit. Currently the most efficient construction of a PRG from a distributional OWF first goes from a distributional OWF to a *weak* OWF, which in turn implies a PRG. On the other hand, obtaining computational entropy from distributional primitives directly would mean skipping the intermediate steps. Furthermore, currently distributional one-way functions are mainly used as a stepping stone to establish standard one-way functions, but can we construct these primitives *directly* from distributional hardness? How efficient are such direct constructions, and can computational entropy be useful in this context? How much entropy can we obtain from distributional hardness, and how tightly does this relate to the entropy obtainable from "standard" hardness?

dOWF to PRG. We consider the current best black box construction of PRGs from OWFs due to Vadhan and Zheng, where a standard one-way function (invertible with probability at most γ) can be shown to have $\log\left(\frac{1}{\gamma}\right)$ relative pseudoentropy, i.e.

$$\mathsf{KL}_{x \leftarrow\$ \{0,1\}^n}(f(x), x \| f(x), \mathcal{A}(f(x))) > \log \frac{1}{\gamma}.$$

for all probabilistic polynomial time adversaries $\mathcal{A}$. We show that *the same* bound is achievable when f is only *distributionally* one-way, where the lower bound on the achievable *statistical distance* is $1 - \gamma$.

From the bound on relative entropy, Vadhan and Zheng further show in particular that for a function that is invertible with negligible probability (i.e. $\gamma = \mathsf{negl}(n)$) on inputs of length n, it holds that there exists a PRG with seed length

$$s = \mathcal{O}\big(n^3 \log n\big).$$

Although Vadhan and Zheng note that relative pseudoentropy also generalizes to distributional one-way functions, in this thesis we show that the relative pseudoentropy obtainable from merely a distributionally-one way function is asymptotically *equivalent*, thus yielding an identical construction of PRGs with identical parameters from the much more general assumption of distributional one-way functions. Although PRGs and distributional OWFs were already known to be existentially equivalent, the fact that the bounds and resulting construction are in fact *equivalent* is of conceptual interest. Namely, the construction does not require one-wayness to obtain the

optimal seed length, and distributional hardness suffices. Furthermore, this yields the current best known conversion of (strong) distributional OWFs to strong OWFs, as the current best conversion in literature is seemingly still the original result die to Impagliazzo and Luby [22]. Impagliazzo and Luby show that a $\frac{1}{n^c}$ distributional OWF for any constant c yields a $1 - \frac{1}{n^{4c+10}}$-weakly one-way function, with inputs of length $\mathcal{O}(n)$. Even for a strong distributional OWF this incurs a cost of n^{-10}.

We establish our results via information theoretic arguments from the distributional hardness bound on statistical distance, and relating this to relative entropy via Bretagnolle-Huber's inequality. Relative entropy can then further be used to obtain next-block pseudoentropy, which can then further be used to obtain a PRG.

dCRHF to SHC. Additionally, we consider the construction of statistically hiding commitments from distributional collision resistance due to BHKY, using the construction of staistically hiding commitments (SHCs) from inaccessible entropy due to Haitner, Reingold, Vadhan and Wee, and provide an explicit analysis of the number of parallel rounds needed in the construction of BHKY. Also here, we show that by using Bretgnolle-Huber's inequality we may tighten the bound on inaccessible entropy from distributional collision resistance established by BHKY. Specifically, BHKY obtain that the generator $G(h, x) = (h(x), x)$ has accessible entropy at most $n - \frac{1}{4p^2}$, where p is the positive polynomial bounding the statistical distance of h. Using Bretagnolle-Huber instead of Pinsker, we improve this bound to accessible entropy at most $n - \log(p)$ (ignoring constants). This in turn also implies a much smaller number of needed parallel instances in the construction of SHCs.

Outline Chapter 2 covers both mathematical and information theoretical definitions and preliminaries, as well as necessary cryptographic definitions. Chapter 2 also covers the aforementioned transformations of computational entropies (Section 2.3) which make up the central lemmas for understanding the actual constructions for sake of brevity. We will only sketch the proofs of the transformation lemmas fo computational entropy. Chapters 3 and 4 include presentations of both the constructions of PRGs and SHCs respectively from computational entropy as well as a summary of the main arguments, adding to the few existing unified treatments of accessible and pseudoentropy (see Haitner and Vadhan [20] and Vahan [36]). Chapters 3 and 4 also contain the main novel contributions and analysis with regards to constructions from distributional hardness. We omit the equivalence of

next-block pseudoentropy and relative entropy due to Vadhan and Zheng, and the full proof of an inaccessible entropy generator yielding a SHC will due to their complexity. The final chapter seeks to contextualize the results of Chapter 3 and Chapter 4, and points out open questions and points we feel the thesis raises, tbuilding on the technicl content established in the chapters before.

Chapter 2

Preliminaries and definitions

Notations and conventions Throughout this thesis we are concerned with random variables drawn from some distribution over bitstrings. For simplicity we thus always assume a discrete probability space. The length (measured in the number of bits) of a string x is denoted $|x|$. For a set S the notation $|S|$ refers to its cardinality. The support of a random variable is defined as the set $\{x \in \Omega : \Pr[X = x] > 0\}$ and is denoted by $\mathrm{Supp}(X)$.

Given two strings x and y we write x, y or (x, y) for a tuple of the strings and $x_{<i}$ for the bits from 1 o $i-1$ of x (with $x_{\leq i}$ being analogous). We say a random variable X is and m-block random variable if it is defined as a sequence of m *blocks* of strings, i.e. $X = (X_1, X_2, ..., X_m)$, where $|X_i| \geq 1$ for all i. For example, we can see the random variable $B = (b_1, b_2)$ as one random variable consisting of two bits, or as a two-block random variable with two single bit blocks. The i-th block of X is denoted by $X[i]$, and the $i-1$ first blocks of X are denoted by $X[< i]$.

Sampling an element x uniformly at random from $\mathcal{D}$ is denoted by $x \leftarrow\!\!\$\, \mathcal{D}$. For example sampling a uniformly random bit $b \leftarrow\!\!\$\, \{0,1\}$ means that b takes either the value 0 or 1 with equal probability, whereas $b \leftarrow\!\!\$\, \{0,1\}^n$ means b takes any of the 2^n possible values for bitstrings of length n with probability equal to 2^{-n}. We use the Kleene star to denote the set of all bitstrings (of unspecified length) $\{0,1\}^*$. This notation will only be used in definitions of cryptographic primitives that need to work with arbitrary length inputs and outputs, as opposed to being defined for some fixed domain and range. Logarithms are always in base 2. For $n \in \mathbb{N}$ the set of integers ranging from 1 to n is denoted by $[n] = \{1, ..., n\}$.

We use $\mathcal{O}(\cdot), \Omega(\cdot), \omega(\cdot)$ in their standard asymptotics meaning of a tight upper bound, tight lower bound and non-tight lower bound for a given function. A function is *negligible* if it tends to zero faster than any positive inverse polynomial (i.e. the function is of the class $n^{-\omega(1)}$), and use $\mathsf{negl}(n)$ to denote

an arbitrary negligible function.

We say the reduction from (the security of) a primitive A to (the security of) primitive B is *black-box* if it only uses the input-output behaviour and security properties of A, and assumes nothing about the structure (or "code") of A. Black box constructions allow us to consider the relationship between the *properties* of two primitives, rather than comparing structures of actual functions or algorithms, which is much harder to generalize. For a rigorous definition of the notion of black box reductions, cf. *Notions of Reducibility between Cryptographic Primitives* by Reingold, Trevisan and Vadhan [32].

2.1 Cryptography

Complexity based cryptography relies on the idea of *computational infeasibility*, in that an adversary should be unable to break some security property given only finite computational resources. The running time of all functions, protocols and adversaries is measured as a function of an integer known as the *security parameter* n. All algorithms receive the security parameter as input, encoded as a string 1^n of n ones. Encoding the security parameter in unary is a practice originating from complexity theory where the complexity of an algorithm is defined as a function of its input length. All adversarial algorithms are assumed to run in probabilistic polynomial time (PPT), and are always uniform. We think of the adversary as a uniform algorithm receiving the security parameter as an input, and consider the output as a random variable.

Definition 2.1.1 (OWF)**.** *We say that an efficiently computable function $f : \{0,1\}^* \to \{0,1\}^*$ is $\delta(n)$-one-way for some function $\delta := \delta(n)$ if for every probabilistic polynomial time adversary $\mathcal{A}$ it holds for all large enough n that*

$$\Pr_{x \leftarrow\$ \{0,1\}^n}\left[\mathcal{A}(f(x)) \in f^{-1}(f(x))\right] < \delta.$$

If f is $\mathsf{negl}(n)$-one-way we say it is a strong *one-way function. When f is $\frac{1}{p}$-one-way for some postivie polynomial p it is $(\frac{1}{p})$-weakly-one-way.*

Definition 2.1.2 (PRG)**.** *An efficiently computable function $G : \{0,1\}^* \to \{0,1\}^*$ is a pseudorandom generator if it has positive stretch s, i.e. $s(|x|) := |G(x)| > |x|$ for all large enough lengths of x, and its output is indistinguishable from the uniform distribution, i.e. for every PPT distinguisher $\mathcal{D}$ it holds that the advantage $\mathsf{Adv}_{\mathcal{D},\mathsf{PRG}}(n) :=$*

$$|\Pr_{y \leftarrow\$ \{0,1\}^{s(n)}}[\mathcal{D}(y, 1^n) = 1] - \Pr_{x \leftarrow\$ \{0,1\}^n}[\mathcal{D}(G(x), 1^n) = 1)]|$$ *is negligible in n.*

2.1.0.0.1 Commitment schemes A commitment scheme is a protocol where a sender S *commits* to a secret message m (in our case a single bit) and sends this commitment c to a receiver R. Separately, in the so-called reveal (or decommitment) phase, S *reveals* the original message (and possibly some additional randomness used in the commitment) and R can check the revealed message to be certain that the message is unaltered from what was originally committed to. In other words, S is *bound* to the original value by the commitment z. In turn S can be certain that until revealing the secret message m, R had no information about the original message m (i.e. the message is *hidden* from R).

We define the notion of a statistically hiding and computationally binding commitment scheme similarly to the formulation by Haiter, Reingold, Vadhan and Wee [19]. More formally, a commitment scheme is a two party protocol $\mathsf{Com} = (\mathsf{S}, \mathsf{R})$ consisting of two phases:

On common security parameter 1^n:

Commit: The sender S commits to its private input b using some internal randomness r. At the end of the phase both parties (S, R) have a common commitment value c, and S has the triple (b, r, c).

Reveal: Both parties receive as input a commitment c, and S receives its original input b and the internal randomness used in the commitment stage. The parties interact, with S revealing the message, after which the receiver R accepts or rejects message-commitment pair.

The notion of a commitment scheme can either be viewed simply as a pair of algorithms, or as an interactive protocol at the end of which the receiver either accepts or rejects the interaction. We further define the security of a commitment scheme as follows:

Definition 2.1.3 (Statistically hiding (bit) commitment scheme)**.** *A commitment scheme is statistically hiding and computationally binding if it holds that*

Hiding: *For $b = 0$ and $b = 1$, the distributions $\mathsf{Com}(b, 1^n)$ returned by the sender at the end of the commitment phase are statistically indistinguishable (i.e. indistinguishable for even unbounded adversaries) by any receiver R.*

Binding: *For any PPT sender S which can find two triples $(b, r, c), (b', r', c)$ where b' is the negation of the input bit, the probability that the receiver accepts the false bit b' in the reveal phase is negligible.*

The hiding and binding properties can be said to hold both for computationally bounded and unbounded adversaries, in which case the commitment scheme is said to be *computationally* or *statistically* binding/hiding respectively. It is not possible to achieve a statistically hiding *and* binding commitment scheme, as the scheme being statistically binding implies that the commitment c determines the message m information theoretically, while statistically hiding implies that given c all messages m are equiprobable.

We will also use the notion of a *weakly binding* commitment scheme, which is defined analogously, except for the fact that the success of the sender against the binding property is at most $1 - \Theta(\frac{1}{p})$ for some polynomial $p(n)$.

2.1.0.0.2 Hash functions A *hash function* is a function which maps arbitrary sized inputs to fixed length outputs (usually compressing). In both constructions central to Chapter 4 and 5 we require sampling of a random hash function from a family of functions with some useful mathematical property. Specifically we use the notions of k-wise independent hash families, universal hash families and universal one-way hash functions (UOWHFs).

Definition 2.1.4 (k-wise independent hash function family)**.** *Let $\mathcal{H}$ be a set of hash functions $\mathcal{H} = \{h : \{0,1\}^d \to \mathcal{R} \subseteq \{0,1\}^r\}$. We say $\mathcal{H}$ is a k-wise independent hash function family if it holds that*

$$\Pr_{h \leftarrow\$ \mathcal{H}}[(h(x_1), ..., h(x_k)) = (y_1, ..., y_k)] = \frac{1}{|\mathcal{R}|^k},$$

for distinct $(x_1, .., x_k) \in \{0,1\}^d$ and any $(y_1, ..., y_k) \in \mathcal{R}^k$, i.e. $(h(x_1), ..., h(x_k))$ is uniform on its range.

Definition 2.1.5 (Universal hash function family)**.** *Let $\mathcal{H}$ be a set of hash functions $\mathcal{H} = \{h : \{0,1\}^d \to \mathcal{R} \subseteq \{0,1\}^r\}$. We say $\mathcal{H}$ is a universal hash function family if it holds that for distinct pairs $(x_1, x_2) \in \{0,1\}^d$ two distinct inputs on a random hash function h collide with low probability, i.e.*

$$\Pr_{h \leftarrow\$ \mathcal{H}}[h(x_1) = h(x_2)] \leq \frac{1}{|\mathcal{R}|}.$$

Note that if $\mathcal{H}$ is a 2-wise independent hash function family, then it is also a universal hash function family. On the other hand, a universal hash function family is not necessarily 2-wise independent, if for example the inequality in Definition 2.1.5 is strict and h is compressing.

Definition 2.1.6 (UOWHF, [24])**.** *An efficient function family $\mathcal{H} = \{\mathcal{H}_n\}_{n \in \mathbb{N}}$, where $\mathcal{H}_n = \{h : \{0,1\}^{i(n)} \to \{0,1\}^{l(n)}\}$ is called a universally one-way hash*

function family if it is compressing, and it is hard to find a collision for a random hash function h on a pre-image chosen by the adversary independently of the hash function, i.e. for any two-stage PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ the probability

$$\Pr[h(x) = h(x') \text{ such that } x' \neq x \text{ when } x \leftarrow\!\!\$\, \mathcal{A}_1(1^n) : h \leftarrow\!\!\$\, \mathcal{H} : x' \leftarrow\!\!\$\, \mathcal{A}_2(x, h)]$$

is negligible in n.

Note that in Defintion 2.1.6 the function family is defined as a collection of families indexed by the security parameter n, where each family is defined by a collection of functions from i to l bits. Thus when sampling a hash function h we first choose the correct family $\mathcal{H}_n$, and then sample a random function.

It is known (cf. [24]) how to construct UOWHFs from arbitrary one-way functions. As such, since the thesis considers constructions of primitives from one-way functions (or the stronger notion of distributional collision resistance), we require no additional complexity assumptions for the existence of UOWHFs. In this thesis the exact efficiency of this construction is not meaningful as universal UOWHFs are merely used as a part of the construction of SHCs, we merely underline the fact that this use is justified since we already assume the existence on OWFs.

2.2 Information Theory

In this section we state Jensen's inequality and the chain rule of relative entropy, and introduce the notions of min- and max-entropy, as well as next-block pseudo min-entropy.

Recall that a twice differentiable real-valued function is convex if and only if its second derivative is non-negative, and that a function is concave if and only if it is the negative of a convex function.

Lemma 2.2.1 (Jensen's inequality)**.** *Let X be a random variable and f be a convex function. Then it holds that*

$$\mathbb{E}[f(X)] \geq f(\mathbb{E}[X]).$$

Conversely for a concave function g it holds that $\mathbb{E}[g(X)] \leq g(\mathbb{E}[X])$

In addition to the chain rule defined in Section 1.1, we also use the chain rule of *relative entropy*.

Lemma 2.2.2. *For random variables (X, Y) and (X', Y') it holds that*

$$\mathsf{KL}((X,Y)\|(X',Y')) = \mathsf{KL}(X\|X') + \mathbb{E}_{x \leftarrow\$ X}\left[\mathsf{KL}(Y\|Y')_{|x_1}\right],$$

where $\mathsf{KL}(Y\|Y')_{|x_1}$ denotes the conditional divergence

$$\mathsf{KL}((Y|X = x_1)\|(Y'|X' = x_1)).$$

A random variable X where $\Pr[X = x] = 0.999$ for some x can still have "high" Shannon entropy with a large enough support, even though the random variable itself is almost constant. The notions of *min-entropy* is used as a more conservative estimate of uncertainty compared to Shannon entropy. The notion of *max-entropy* of a random variable X is defined via the size of the support of X.

Definition 2.2.1 (Min- and max-entropy)**.** *The min-entropy of X is denoted by $\mathrm{H}_\infty(X)$ and is given by*

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

We define the max-entropy of X by $H_0(X) = \log |\operatorname{Supp}(X)|$.

Intuitively high min-entropy means all values of X have small probability mass, and low max-entropy means the random variable has small support. One can show that in general it holds

$$\mathrm{H}_\infty(X) \leq \mathrm{H}(X) \leq H_0(X)$$

with equality when X is uniform on its support. Conditional min-entropy is defined analogously to conditional Shannon entropy, as originally due to Dodis, Ostrovsky, Reyzin and Smith [11]

Definition 2.2.2. *For jointly distributed random variables (Y, X) we define the conditional min-entropy of X given Y as*

$$\mathrm{H}_\infty(X|Y) = \log\left(\frac{1}{\mathbb{E}_y[\max_x \Pr[X = x|Y = y]]}\right)$$

Conditional pseudo-min-entropy is defined analogously to pseudoentropy:

Definition 2.2.3 (Next-block pseudo min-entropy)**.** *An m-block random variable $X = (X_1, ..., X_m)$ has next-block pseudo min-entropy (at least) k for each block if it holds that for $i \in [m]$ each X_i is computationally indistinguishable by oracle aided distinguishers from some Y_i when conditioned on previous blocks $X_1, ..., X_{i-1}$, where $Y = (Y_1, ..., Y_m)$ is jointly distributed with X and that*

$$\mathrm{H}_\infty(Y_i|X_{\leq i-1}) \geq k.$$

for all $i \in [m]$

The crucial difference between Definition 2.2.3 and 1.1.6 is that Definition 2.2.3 specifically captures that *every block* appears to have min-entropy at least k.

When using pseudoentropy notions, we thus far omitted that the uniform distinguishers need to have an oracle that allows for efficient sampling from joint distributions in the nbpe definitions. For all adversarial algorithms against pseudoentropy we must implicitly assume such an oracle against the random variable and the joint indistinguishable variable, as given in Definition 2.2.3. We generally omit this technical detail in the exposition. Alternatively one can formulate very similar results by considering nonuniform adversaries with the samples as nonuniform advice (where we instead consider our adversarial algorithms as families of circuits indexed by the length of the input).

2.3 Transformations of computational entropy

We devote this final section to entropy transformation lemmas introduced in the context of PRG constructions by Haitner, Reingold and Vadhan [18] and commitment scheme constructions by Haitner, Holenstein, Reingold, Vadhan and Wee [19].

A central property of both pseudoentropy (Def. 1.1.6) and accessible entropy (Def. 1.1.9) is that known information theoretic transformations of entropy, such as taking independent copies, also yields similar results for computational entropies. More specifically, given an m-block random variable with some computational entropy we may *equalize* the entropy over the blocks via random truncation and serial repetition, *flatten* Shannon entropy to min-entropy via parallel repetition, and use hash functions on high min-entropy variables to obtain a close to uniform output. This means that we can go from a bound on the total computational entropy to high computational min-entropy such that all strings have "low" probability mass. This in turn means applying a hash function will yield a relatively even output distribution, a fact that is used in both constructions in Chapter 3 and 4 respectively. Recall that we defined pseudo-min-entropy analogously to next-block pseudoentropy (Def. 2.2.1).

We also recall from Section 2.1 that the distinguishers against pseudoentropy are assumed to be oracle aided, in that they have access to an oracle sampling from the joint distribution (X, Y), where Y is the random variable achieving the pseudoentropy of X (recall Def. 1.1.6).

The first problem we will need to deal with when using variables with

some form of computational entropy is that even though the *total* entropy of a random variable may be known, it is not in general possible to know the entropies of individual blocks. For example, the first 1, 10 or 100 bits of any variable could be trivial zeros depending on the application, and thus have zero entropy. Since estimating entropy thresholds is expensive (cf. Valiant and Valiant [38]), we instead opt to *equalize* the entropies. In entropy equalization, we define a new random variable via taking w independent copies of an m-block variable X and removing the first j blocks of the first copy of X and last $m - j$ blocks of the last copy for a uniformly random $j \in [m]$, leaving us with $m \cdot (w-1)$ blocks. The effect this has is that each block of the original random variable is now uniformly distributed in the resulting X^w. It is then possible to show (cf. [18]) that the entropy of each individual block is simply the average of the original total entropy (real or computational).

Lemma 2.3.1 (Equalization Lemma)**.** *Let $X = (X_1, ..., X_m)$ be an m-block random variable with. Define the $(w-1) \cdot m$-block random variable X^w by $(X_1[j], ..., X_w[j-1])$ where $X_i[j]$ denotes the j-th block of the i-th independent copy of X and j is a uniformly random element of $[m]$ (see Fig. 2.1 below). Then conditioned on j and previous blocks it holds that:*

1. ***Real entropy*** *If X has total real entropy at least k, each block of X^w has real entropy at least k/m.*
2. ***Pseudoentropy*** *If X has total pseudoentropy at least $k+\Delta$, each block of X^w has next-block pseudoentropy at least $\frac{k+\Delta}{m}$.*
3. ***Accessible entropy*** *If X has total accessible entropy at most $k - \delta$, X^w has accessible entropy at most $(w-2) \cdot (k-\delta) + 2 \cdot H_0(X) + \log m$ where each X is the output of an efficient m-block[1] generator G on independent uniformly random inputs.*

We include a visualization of the concrete construction of X^w in Figure 2.1 below:

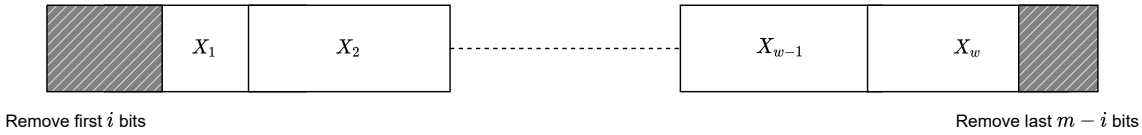

Figure 2.1: Entropy Equalization

[1]Here m is assumed to be a power of two to allow for efficient construction of G^w.

Proof sketch. Let X be an m-block random variable with entropy bounds as given in Lemma 2.3.1. The bound on real entropy and next-block pseudoentropy follow by almost identical arguments, while the accessible entropy proof is very different.

Real entropy For the real entropy we use Lemma 1.1.1, as conditioning on more information can only decrease the entropy. This implies the inequality

$$\mathrm{H}(X^w[i]|X^w[<i]) \geq \mathrm{H}\Big(\vec{X}[i+j-1]|\vec{X}[<i+j-1], j\Big),$$

where $\vec{X} = (X_1[1], ..., X_w[m])$ is the untruncated version of X^w. The proof then proceeds by writing the conditional entropy as the expectation over j, and the fact that the entries are circular, meaning $\vec{X}[i+j-1] = X[i+j-1 \mod m]$ in distribution. Then, since $i' := i+j-1 \mod m$ is uniform in $[m]$ we obtain bounds on the expectation

$$\mathbb{E}_{i' \mod m \leftarrow\$ [m]}[\mathrm{H}(X[i']|X[<i'])] = \frac{1}{m}\sum_{i' \in [m]} \mathrm{H}(X[i']|X[<i']),$$

and similarly for $Y[i']$ in place of $X[i']$. We note that in the case that X is an m-block generator HRVW further condition on a possible public seed parameter given to the generator, but this has no effect on the proof.

Pseudoentropy The pseudoentropy version is analogous to the real entropy case, except we estimate the conditional entropy of Y^w, where Y is the variable that realizes the next-block pseudoentropy of X.

Accessbile entropy For the accessible entropy we assume that there exists some G^w-consistent simulator $\widetilde{G^w}$ that contradicts the bound on the accessible entropy. From this it is possible to show that a random substranscript (i.e. a random input of the simulator against G^w) will have accessible entropy greater than $k-\delta$. It follows that outputting the blocks of this subtranscript will be both in the support of the original generator G, and achieves accessbile entropy greater than $k-\delta$, contradicting the accessible entropy if G. □

Recall that to get meaningful lower bound on entropy, such that all possible values have low probability mass, we need the notion of *min-entropy* (see Def. 2.2.1). This process of obtaining min-entropy from Shannon entropy is done via parallel amplification, where we take a t-fold parallel product of the original random variable.

Lemma 2.3.2 (Flattening Lemma)**.** *Let X be an m-block random variable $X = (X_1, ..., X_m)$, such that each block is of length l. Let $X^{<t>}$ be the t-fold parallel repetition of X, such that the i-th block equals the t-fold parallel repetition $X^{<t>}[i] = (X^1[i], ..., X^t[i])$, where X^i are i.i.d. Then it holds that:*

1. ***Real min-entropy*** *If* ***each block*** *of X has real entropy k, then the blockwise real min-entropy of $X^{<t>}$ is at least*

$$t \cdot k - \mathcal{O}\Big((\log n + l) \cdot \log n + \sqrt{t}\Big)$$

2. ***Pseudo min-entropy*** *If* ***each block*** *of X has real entropy k_{pseudo}, then the blockwise pseudo min-entropy of $X^{<t>}$ is at least $t \cdot k_{pseudo} - \mathcal{O}\big((\log n + l) \cdot \log n + \sqrt{t}\big)$*

3. ***Accessible entropy*** *If X has total accessible entropy at most k_{acc}, then the total accessible entropy is at most $t \cdot k_{acc}$, when X is the output of an m-block generator G.*

We include a visualization of the concrete construction of $X^{<t>}$ in Figure 2.2 below:

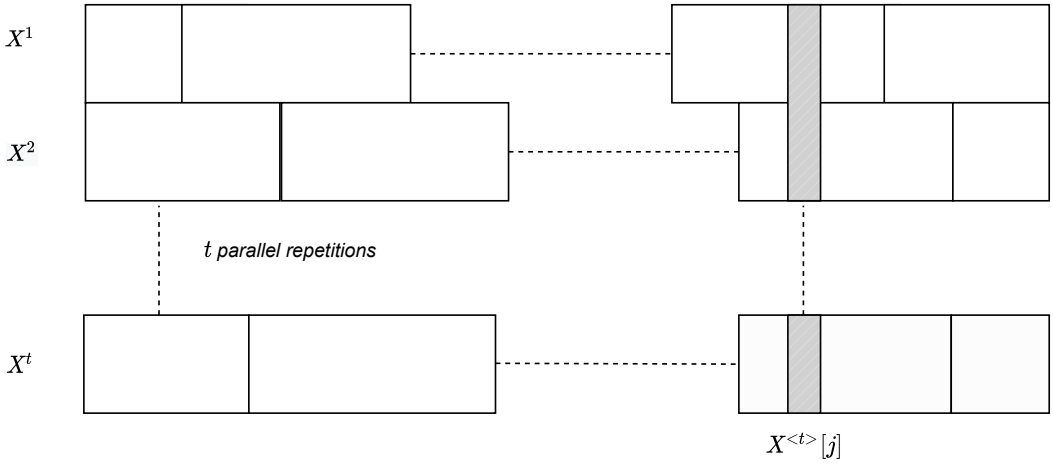

Figure 2.2: Entropy Flattening

Proof sketch. Let X be an m-block random variable with entropy bounds as given in Lemma 2.3.2, and Y be the joint random variable realizing the pseudoentropy of X.

Real min-entropy The bounds on real and pseudoentropy again follow in very similar fashions. Define the *sample entropy* of a random variable X as $H_X(x) = \log\left(\frac{1}{\Pr[X=x]}\right)$, i.e. the entropy functional of a *single* realization of X. The sample entropy of $X^{<t>}$ (and $Y^{<t>}$ conditioned on previous blocks of X) can be bounded such that with probability at least $1 - \varepsilon$ over a random

realization of $x \leftarrow\!\!\$\, X^{<t>}[1,...,i]$ (and $(X^{<t>}[<\ i], Y^{<t>}[i])$ respectively) it holds that

$$H_{X^{<t>}}(x) - t \cdot \mathrm{H}(X) \geq -\mathcal{O}\Big((\log n + l) \cdot \log n + \sqrt{t}\Big).$$

The above bound can be established via concentration bounds on the sample entropy, cf. Theorem 3.14 of Guang [41]. Given the bound

$$H_{X^{<t>}}(x) - t \cdot \mathrm{H}(X) \geq -\mathcal{O}\Big((\log n + l) \cdot \log n + \sqrt{t}\Big).$$

it is then straightforward to set $\varepsilon = 2^{-(\log n)^2}$ to obtain the bound on real entropy.

Pseudo min-entropy For $Y^{<t>}[i]$ we further need the fact that $(X^{<t>}[< i], Y^{<t>}[i])$ is ε-close (in statistical distance) to some random variable $(X^{<t>}[< i], W[i])$ that *achieves* the min-entropy (such that the concentration bound holds with probability 1), i.e. there exists a random variable achieving the claimed pseudo-min entropy of $(X^{<t>}$ for each block. The closeness of $(X^{<t>}[< i], Y^{<t>}[i])$ to $(X^{<t>}[< i], W[i])$ follows from the fact that the concentration bound holds with probability at least $1 - \varepsilon$.

The proper reduction against the next-block pseudo min-entropy then proceeds by distinguishing $(X[< i], X[i])$ from $(X[< i], Y[i])$ by sampling a realization of $(X^{<t>}[< i], Y^{<t>}[i])$ (recall that our distinguishers are assumed to be oracle aided w.r.t sampling), and replacing a random prefix of a single row with the candidate value $((X[< i], Z))$, and using the distinguisher against the claimed min-entropy. By a standard hybrid argument this then yields a distinguisher against the original blockwise pseudoentropy.

Accessible min-entropy The bound on the accessible entropy follows by assuming an efficient simulator $\widetilde{G}$ contradicting the accessible entropy, which then implies that, as the total accessible entropy is at least $t \cdot k_{acc}$, a random column $\widetilde{G}[i] = (\widetilde{X}^1[i], ..., \widetilde{X}^t[i])$ of the simulator has to have more than k_{acc} accessible entropy. Otherwise, the average accessible entropy will be less than $t \cdot k_{acc}$. This then further implies an efficient consistent generator against the original accessible entropy bound, by simply choosing a random $j \in [t]$ and outputting the j-th column of the simulator $\widetilde{G}$. This can then be shown to contradict the bound on the original accessible entropy. □

The above conversion from Shannon to min-entropy is essentially a quantitative version of the *asymptotic equipartition property* from information theory (cf. [10]). In essence when taking a large number of independent samples of a random variable, the joint variable will be in a so-called *typical set* with probability $1 - \varepsilon$, with all probabilities in the typical set being

roughly equal. This can then be directly translated to a bound on the min-entropy, as with high probability all values have "equal" probability mass. Note that the necessary number of independent samples are a source of huge loss in all constructions from computational entropy due to going from Shannon to min-entropy. On the other hand Chen, Göös, Vadhan and Zhang [9], show that $\Omega(n^2)$ invocations to the original random variable are *necessary* for black-box entropy flattening.

We note that the randomness of j in Lemma 2.3.1 can actually be voided, if Lemma 2.3.1 is applied together with Lemma 2.3.2. This is due to the fact that if we instead equalize the variables deterministically from $j = 1$ to $j = m$ each block will appear an equal number of times in each parallel variable. This means that the total min-entropy will again be exactly the average. We will turn to this in Chapter 3.

We conclude by the block source extraction lemma that allows us to obtain (computationally indistinguishable from) uniform random bits given a bound on the pseudo-min-entropy. The heart of Lemma 2.3.3 relies in the Leftover Hash Lemma due to Håstad, Impagliazzo, Levin and Luby [21], which roughly states that a random variable with sufficient min-entropy will yield close to uniformly random bits when hashed. This then also yields computational randomness (i.e pseudorandomness) — namely if the computational min-entropy does not yield indistinguishably random bits, then the original random variables are distinguishable, contradicting the pseudo-min-entropy. The central idea is that using a hash function on a random variable with some min-entropy allows us to obtain a string that appears uniformly distributed:

Lemma 2.3.3 (Block source extraction)**.** *Let $X = (X_1, ..., X_m)$ be an m-block random variable such that **each block** of X has pseudo-min-entropy (at least) k_{pseudo}. Then for $Ext(h, X) = (h, h(X_1), ..., h(X_m))$, where h is a random hash function mapping strings of length t to strings of length $\lfloor k_{pseudo} \rfloor - \lceil \log^2 n \rceil$, it holds that*

$$\Pr[\mathcal{D}(h, Ext(h, X), 1^n) = 1] - \Pr\left[\mathcal{D}(h, U_{m\cdot|h|}, 1^n) = 1\right] < \mathsf{negl}(n)$$

for any oracle aided PPT distinguisher $\mathcal{D}$, where $U_{m\cdot|h|}$ denotes a uniformly random string of length $m \cdot |h|$.

The exact nature of the hashing in relation to the previous manipulations is illustrated below in Figure 2.3.

Proof sketch. The proof against the pseudorandomness follows via a standard hybrid argument. The i-th hybrid is defined as

$$H_i(Z) = (h, h(X_1), ..., h(X_{i-1}), h(Z), U_{(k_{pseudo}-\kappa)(m-i)}).$$

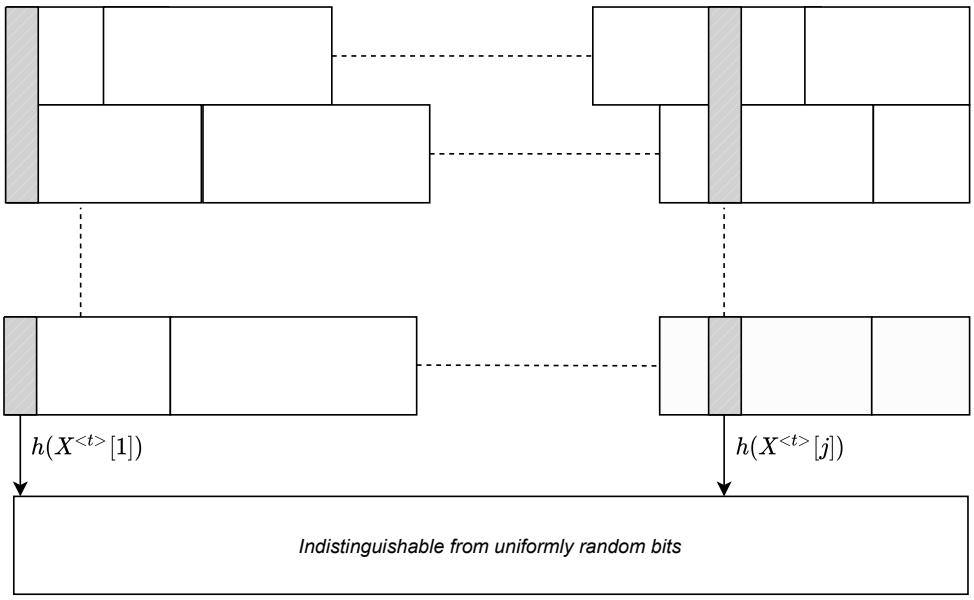

Figure 2.3: Block source extraction

The PRG-advantage will then directly be the extreme hybrids. Two subsequent hybrids can then be bounded by the distinguishing advantage of $H_i(X_{i-1})$ and $H_i(Y_i)$. This is due to the fact that the distance from

$$H_i(Y_i) = (h, h(X_1), ..., h(X_{i-1}), h(Y_i), U_{(k_{pseudo}-\kappa)(m-i)})$$

to

$$H_i(X_{i-1}) = (h, h(X_1), ..., h(X_{i-1}), U_{(k_{pseudo}-\kappa)(m-i+1)})$$

is exactly the statistical distance between $h(Y_i)$ and $U_{(k_{pseudo}-\kappa)}$ by definition. By the leftover hash lemma this distance is at most $2^{-\kappa/2}$. Telescoping over all indices yields that the distinguishing advantage of the extreme hybrids (the PRG advantage) is bounded by m times the distinguishing advantage against X_i and Y_i conditioned on previous blocks, plus $2^{-\kappa/2}$. This then yields that the distinguishing advantage for any PPT distinguisher against the the pseudorandomness of $Ext(h, X)$ is negligible. □

Chapter 3

Pseudorandom generators from distributional OWFs

As mentioned in the introduction, we now to present the current best (in terms of seed length) fully black box construction of a PRG from an arbitrary OWF due to Vadhan and Zheng [37]. After covering the construction and its underlying proof, we will then analyze the more general case when the underlying function is merely *distributionally* one-way. Since distribtuionally one-way functions imply OWF, distributional OWFs also imply PRGs. The exact tightness (and indeed resulting equivalence) under more general functions has not been studied in current literature.

Vadhan and Zheng also note in their original paper that relative pseudoentropy (or KL-hardness for sampling) allows for constructions of PRGs from a much more general class of functions. Although the fact that PRGs can be constructed from dOWFs is in a sense implicit in the original paper, we find the fact that distributional one-way functions allow for an *equivalent* construction highly nontrivial and an observation worthy of interest. In fact, we may take a function that is invertible with probability 1, such as the example in eq. (1.1), and obtain an asymptotically optimal (to current knowledge) PRG, despite having a function with no one-wayness whatsoever.

We summarize the original result of Vadhan and Zheng result as follows:

Theorem 3.0.1 (OWF $\implies$ PRG [37] Corollary 5.8)**.** *Let f be a strong one-way function. There then exists a pseudorandom generator $G : \{0,1\}^s \to \{0,1\}^{2s}$ with seed length*

$$s = \mathcal{O}\big(n^3 \log n\big)$$

making black-box calls to the one-way function f on inputs of length n.

This improves on the original nbpe-based construction of PRGs from OWFs due to Haitner, Reingold and Vadhan [18], which needs a seed of

length $\mathcal{O}(n^4 \log n)$. The factor n improvement in the construction of Vadhan and Zheng is due to using extracted bits as new inputs to our next-block pseudoentropy generator, reducing the number of uniformly random bits needed as input.
We arrive at the theorem via the following steps:

1 - PEG : Show that $(f(x), x_1, ..., x_n)$ is a next-bit pseudoentropy generator (PEG) with $n + \omega(\log n)$ next-bit pseudoentropy (nbpe).

2 - PRG_Z: Establish that the output of a next-bit pseudoentropy generator can be equalized, flattened and extracted to obtain a generator whose distribution that is computationally indistinguishable from the input distribution concatenated with random bits, known as a Z-seeded PRG

3 - PRG: Show that iterating a Z-seeded PRG PRG_Z yields a PRG. The proof is a standard hybrid argument.

Moreover, it suffices to assume that f is distributionally one-way in Step 1 to establish that $G_{nb} = (f(x), x_1, ..., x_n)$ is a next-bit pseudoentropy generator. Indeed, if there is a $(1-\mathsf{negl}(n))$ gap between the uniform distribution and the distribution (measured in total variation distance) produced by any efficient inverter for f (cf. Definition 1.1.10), then the Bretagnolle-Huber bound 1.1.3 allows us to conclude that G_{nb} is a next-block pseudoentropy generator with $n + \omega(\log n)$ total next-bit pseudoentropy.

We will first provide a self contained summary of Theorem 3.0.1 and its central lemmas in Section 3.1, after which we turn to the distributional construction in Section 3.2, which is the main novel technical contribution included in this chapter.

3.1 Vadhan-Zheng PRG

We now formalize the three previously outlined steps that yield the proof of Theorem 3.0.1.

Lemma 3.1.1 (Step 1 - PEG :)**.** *Suppose f is a strong one-way function. Then $G_{nb}(x) = (f(x), x_1, ..., x_n)$ is a next-bit pseudoentropy generator with $n + \omega(\log n)$ total next-bit pseudoentropy.*

To formalize Step 2, we need the following construction of a Z-seeded generator given by Vadhan and Zheng. The generator PRG_Z takes as input a variable z that is strucured as

$$z = (h, (G_{nb}(x^i)_{<i \mod n})_{i=1}^t, (G_{nb}(x^{t+i}))_{i=1}^t),$$

where we denote $X_{<i \mod n} := (X_1, ..., X_{i \mod n-1})$ as the first $i \mod n$ bits (or blocks) of the random variable. In other words we cyclically remove 1 to n bits for the first n realizations of i, after which we again remove 1 to n cyclically until we reach $i = t$. We fix the parameter $t = \mathcal{O}(n^2 \log n)$, which, as we will see in Section 3.1.2, is determined by the necessary number of parallel repetitions needed for sufficient min-entropy to allow for randomness extraction via h. the The random variable z consists of

h: A universal hash function h drawn from a universal hash function family (see Def. 2.1.6) mapping t bit inputs (meaning h is also samplable with t bits) to t' bit outputs (where $t'n - tn = |z| \cdot \log n/n$)[1],

$G_{nb}(x^i)$: t evaluations of the next-bit pseudoentropy generator G_{nb} on uniformly random and independent inputs x^i. For the first t generator outputs the algorithm PRG_Z receives truncated evaluations

$$(G_{nb}(x^i)_{<i \mod n})_{i=1}^{t},$$

where we truncate in order from $i = 1$ to $i = n$ bits cyclically. Without loss of generality we assume t is a multiple of n.

$G_{nb}(x^{t+i})$: t evaluations of the next-bit pseudoentropy generator G_{nb} on uniformly random inputs without truncation.

To sample one realization of the random variable z we need $|h| + 2t|x| = t + 2nt = \mathcal{O}(n \cdot t)$ bits of randomness, where $|h|$ is the descritption length of h. The pseudocode for PRG_Z is presented below.
Note that PRG_Z does not apply equalization identically to the formulation in Lemma 2.3.2, where we would instead sample a random integer $j_r \in [m]$ and remove the first j_r and last $n - j_r$ bits. Instead in the pseudocode above for each row we *deterministically* set $j_r = i = 1$ to $i = n$ cyclically. These two yield equivalent entropy bounds, which formalize as the following claim:

Claim 3.1: *The average blockwise next-bit pseudoentropy of y_i in the code of PRG_Z when truncating the pairs $(G_{nb}^{t+i}, G_{nb}^{i})$ deterministically is equivalent to the blockwise nbpe of y_i if we truncated randomly with a unique $j_r \in [m]$ for each pair.*

[1]The difference is due to known optimality bounds due to Gennaro, Gertner, Katz and Trevisan [14] whose result implies bounds on the possible obtainable stretch per invocation of the underlying OWF.

$\mathsf{PRG}_Z(z, 1^n)$

$(h, ((G^i_{nb})_{<i \mod n})^t_{i=1}, ((G^{t+i}_{nb})^t_{i=1})) \leftarrow$ **parse** z
for $i \in [t]$ **do**
  $y_i \leftarrow (G^{t+i}_{nb})_{i \mod n,...,n, (G^i_{nb})_{<i \mod n}}$
  // set two generator outputs as row and cut off ends
for $j \in [n]$ **do**
  $y'_j \leftarrow (y_1[j], ..., y_t[j])$
  // Set the j-th bits of each row as columns
  $x_j \leftarrow h(y'_j)$ // hash each column
$\hat{x}_1, ..., \hat{x}_t, \hat{x} \leftarrow (x_j)^m_{j=1}$ // parse randomness
return $((h, ((G^{t+i}_{nb})_{<i \mod n}))^t_{i=1}, (G(\hat{x}_i))^t_{i=1}, \hat{x})$

Figure 3.1: Z-seeded generator PRG_Z

Proof sketch of Claim 3.1. The proof of the claim follows by noting that with deterministic offsets the first n bits of each column $(y_1[j], ..., y_n[j])$ will have $n + \Delta$ total bits of next-bit pseudoentropy (as every block appears exactly once, and all blocks in total have nbpe at least $n+\Delta$ by assumption on G_{nb}), meaning the total pseudoentropy in a full column is

$$(n + \Delta) \cdot t/n = t \cdot (n + \Delta)/n.$$

The right hand side is exactly the next-bit pseudoentropy of each column using random truncation (as equalizing yields $(n+\Delta)/n$ bitwise pseudoentropy for each block, multiplied by the number of parallel repetitions t to obtain column-wise pseudoentropy). Thus similarly every block has *on average* the same pseudoentropy as in the random offset case when using deterministic offsets, yielding equivalent min-entropy bounds when invoking Lemma 2.3.2. □

Due to the convenient modularity of the random truncation we will still use the Equalization Lemma in our analysis, as it allows us to consider the equalization and flattening steps as separate entropy manipulations. We present the deterministic version as pseudocode to simplify notation and due to the smaller sampling complexity (as we avoid the necessary $t \cdot \log m$ bits needed to sample all random j_r). [2]

[2]Equivalently the reader may disregard the deterministic version and replace each $i \mod n$ with a unique random $j_r \in [m]$ at each row.

The analysis of construction PRG_Z very closely follows Lemmas 2.3.1, 2.3.2, and 2.3.3 as mentioned earlier. Heuristically the idea is to equalize, flatten and extract our original next-bit pseudoentropy into pseudorandomness, and use some extracted bits as inputs to the underlying next-bit pseudoentropy generator G_{nb}. This allows one to effectively skip sampling all the necessary random bitstrings needed for entropy equalization, as the rows are then instead filled by extracted bits, rather than by needing to provide them all as input. Since the amount of extractable bits in the Extraction Lemma (2.3.3) is effectively a function of t, n and the pseudoentropy gap Δ, it suffices to choose t large enough to obtain sufficient min-entropy, and thus enpugh pseudorandom bits.

We claim that output of PRG_Z is indistinguishable from its own input distribution *concatenated* with uniform randomness. Note that if the input is uniformly random this is exactly the definition of a PRG.

Lemma 3.1.2 (Step 2 - PRG_Z:)**.** *Suppose G_{nb} is a next-bit pseudoentropy generator with $n + \omega(\log n)$ pseudoentropy. Then PRG_Z is a Z-seeded PRG, i.e. for all PPT distinguishers $\mathcal{D}$ it holds that*

$$|\Pr_{z \leftarrow\$ Z}[\mathcal{D}(\mathsf{PRG}_Z(z), 1^n) = 1] - \Pr_{z' \leftarrow\$ Z}[\mathcal{D}((z', u_\sigma), 1^n) = 1]| < \mathsf{negl}(n),$$

where u_σ is a uniformly random string of length $\sigma = |z| \cdot \log n/n$.

Iterating the Z-seeded generator "enough" times will then yield a PRG, as each round we run PRG_Z we obtain some extra bits of randomness, and use the the rest to initialize a new call to PRG_Z.

To obtain the claimed stretch of Theorem 3.0.1 we set the number of iterations l as $l := 2n/\log n$:

Lemma 3.1.3 (Step 3 - PRG:)**.** *Let PRG_Z be a Z-seeded PRG defined as in Lemma 1.1.2, i.e. for all PPT $\mathcal{D}$ it holds that,*

$$|\Pr_{z \leftarrow\$ Z}[\mathcal{D}(\mathsf{PRG}_Z(z), 1^n) = 1] - \Pr_{z' \leftarrow\$ Z}[\mathcal{D}((z', u_\sigma), 1^n) = 1]| < \mathsf{negl}(n),$$

where u_σ is uniform bitstring of length σ and Z is a distribution efficiently samplable with s random bits. Then the function $G_l : \{0,1\}^s \to \{0,1\}^{l\sigma}$ is a pseudorandom generator, where $G = \mathsf{PRG}_Z$ and

$$G_k(z) = (G_{k-1}(z'), u_\sigma)),$$

when $G(z) = (z', u_\sigma)$ and $l = 2n/\log n$.

Proof of Theorem 3.0.1. Given Lemmas 3.1.1 — 3.1.3, we proceed by establishing Theorem 3.0.1. Suppose f is a strong one-way function secure on inputs of length n, and define the next-bit pseudoentropy generator G_{nb} as

$$G_{nb} := (f(x), x_1, ..., x_n).$$

It follows by Lemma 3.1.1 the generator G_{nb} is a next-bit pseudoentropy generator with $n+\omega(\log n)$ total next-bit pseudoentropy. Further, by Lemma 3.1.2 the construction PRG_Z using G_{nb} is a Z-seeded generator, taking as input a variable z distributed according to

$$Z = (h, (G_{nb}(x^i)_{<i \mod n})_{i=1}^t, (G_{nb}(x^{t+i}))_{i=1}^t),$$

consisting of a universal hash function h on t bits and evaluations of the pseudoentropy generator G_{nb} for $2t$ i.i.d copies of $x \leftarrow\$ \{0,1\}^n$, where $t = \mathcal{O}(n^2 \log n)$. Consequently the input for the generator PRG_Z is samplable with $|z| = t+2nt = \mathcal{O}(n \cdot t) = \mathcal{O}(n^3 \log n)$ bits of randomness, and its output is computationally indistinguishable from (z', u_σ), where $z' \leftarrow\$ Z$, and u_σ is a uniformly random string of length $\sigma = |z| \cdot \log n/n$. Finally, by Lemma 3.1.3 using PRG_Z iteratively $\ell = 2n/\log n$ times yields a PRG with seed length $|z| = \mathcal{O}(n^3 \log n)$ (required to initialize the first call to the Z-seeded generator PRG_Z) , and output length $l\sigma = (2n/\log n)(|z| \cdot \log n/n) = 2|z|$. □

We devote the rest of Section 3.1 to proving the three main lemmas. Lemma 3.1.1 follows from an equivalence of relative pseudoentropy and next-block pseudoentropy for short blocks, Lemma 3.1.2 is implied by the entropy manipulation lemmas of Section 2.3, and Lemma 3.1.3 is proved via a standard hybrid argument.

3.1.1 One-way function to next-bit pseudoentropy

To prove Lemma 3.1.1 we recall three Lemmas presented in [37] [1]. First we obtain relative pseudoentropy from a OWF.

Lemma 3.1.4 (OWF has relative pseudoentropy [37] Lemma 4.2)**.** *For a γ-one-way function f, x has $\log \frac{1}{\gamma}$ relative pseudoentropy given $f(x)$, i.e. for all PPT adversaries $\mathcal{A}$ it holds that*

$$\mathsf{KL}_{x \leftarrow\$ \{0,1\}^n}(f(x), x \| f(x), \mathcal{A}(f(x))) > \log \frac{1}{\gamma}.$$

In particular if f is a strong one-way function it has $\omega(\log n)$ relative pseudoentropy.

The proof of Lemma 3.1.4 follows by supposing towards a contradiction and applying the deterministic binary test where $T(y,x) = 1 \iff y = f(x)$ and the data processing inequality (cf. Cover and Thomas Section 2.8 [10]).

We then use the fact that relative pseudoentropy of a uniformly random block given the previous blocks is the arithmetic mean of the total relative pseudoentropy:

Lemma 3.1.5 (Chain rule for relative pseudoentropy [37] Lemma 4.3.)**.** *Let $X = (X_j)_{j=1}^m$ be an m-block random variable jointly distributed with Y. If X has relative pseudoentropy Δ given Y, then the expected relative pseudoentropy of X_i given $Y, X_1, ..., X_{i-1}$ is Δ/m taken over uniformly random i.*

The final and perhaps most central step in showing Lemma 3.1.1 holds is the fact that relative pseudoentropy is *equivalent* to a gap in next-block pseudoentropy for short (i.e. logarithmic length) blocks, as originally established by Vadhan and Zheng.

Lemma 3.1.6 (Relative pseudoentropy equivalence - Theorem 3.21 [37], Lemma 1.6 [1])**.** *Let (Y,X) be a random variable where $|X| = \mathcal{O}(\log n)$ and Y has polynomial length. Then X has relative pseudoentropy at least Δ given Y if and only if X has conditional pseudoentropy at least $\mathrm{H}(X|Y) + \Delta$ given Y, i.e. for all PPT $\mathcal{A}$ it holds that*

$$(\mathsf{KL}(f(x), x \| f(x), \mathcal{A}(f(x))) > \Delta) \iff \left((Y,X) \overset{c}{\approx} (Y,Z)\right),$$

for some Z jointly distributed with Y, X and $\mathrm{H}(Z|Y) = \mathrm{H}(X|Y) + \Delta$

Proof of Lemma 3.1.1. Suppose f is a strong one-way function and define $G_{nb}(x) = (f(x), x_1, ..., x_m)$, where we split the input $x = (x_1, ..., x_m)$ into logarithmic blocks, such that $|x_i| = \log m$ for $i \in [m]$. We want to show that the generator $G_{nb}(x) = (f(x), x_1, ..., x_m)$ has total next-block pseudoentropy equal to $n + \omega(\log n)$, i.e. that there exists some random variable $Z = (Z_0, ..., Z_m)$ jointly distributed with $G_{nb}(x)$, such that

$$\sum_{i=1}^{m} \mathrm{H}(Z_i | G_{nb}(x)_{<i}) \geq n + \omega(\log n),$$

and Z_i each indistinguishable from the i-th block of $G_{nb}(x)$ conditioned on previous blocks.

By Lemma 3.1.4 the relative pseudoentropy of x given $f(x)$ equals $\log n$ (we drop the asymptotic notion of $\omega(\log n)$ wlog) . By directly applying

Lemma 3.1.5 we then obtain blockwise relative entropy equal to $\Delta/m = \log n/m$ for the i-th block of $G_{nb}(x) = (f(x), x)$ (that is to say x_i), given $(f(x), x_1, ..., x_{i-1})$ for a random index $i \in [m]$. Furthermore the relative pseudoentropy is then equivalent to $\log n/m$ extra bits of pseudoentropy for each individual block in next-block pseudoentropy by Lemma 3.1.6, i.e. $G_{nb}(x)_i|G_{nb}(x)_{<i}$ is computationally indistinguishable from some $Z_i|G_{nb}(x)_{<i}$, where

$$\mathrm{H}(Z_i|G_{nb}(x)_{<i}) = \mathrm{H}(G_{nb}(x)_i|G_{nb}(x)_{<i}) + \log n/m. \tag{3.1}$$

Now for some $Z = (Z_1, ..., Z_m)$ jointly distributed with G_{nb} it holds that

$$\begin{aligned}\sum_{j=1}^{m} \mathrm{H}(Z_j|G_{nb}(x)_{<j}) &= m \cdot \sum_{j=1}^{m} \mathrm{H}(Z_i|G_{nb}(x)_{<i}, i=j)\frac{1}{m} \\ &= m\,\mathbb{E}_i[\mathrm{H}(Z_i|G_{nb}(x)_{<i}, i)] \\ &= m\,\mathbb{E}_i[\mathrm{H}(G_{nb}(x)_i|G_{nb}(x)_{<i}, i) + \log n/m] \\ &= \sum_{j=1}^{m} \mathrm{H}(G_{nb}(x)_i|G_{nb}(x)_{<i}) + \log n = n + \log n\end{aligned}$$

where the second equality is by the definition of conditional entropy and conditioning on a random $i \in [m]$, the third equality by eq. 3.1, and the final equality again by the definition of conditional entropy and by linearity of expectation. Thus $G_{nb}(x) = (f(x), x_1, ..., x_m)$ has total next-block pseudoentropy $n + \log n$, as we wanted to show. □

3.1.2 Next-bit pseudoentropy to Z-seeded generator

We now show that, given a next-bit pseudoentropy generator, the construction PRG_Z is a Z-seeded generator. We do this by showing that given the initial next-bit pseudoentropy over the entire output of G_{nb}, applying equalization, flattening and block-source extraction gives us pseudorandom bits, which in turn yields that for any efficient distinguisher $\mathcal{D}$ the advantage

$$|\Pr_{z \leftarrow\$ Z}[\mathcal{D}(\mathsf{PRG}_Z(z), 1^n) = 1] - \Pr_{z' \leftarrow\$ Z}[\mathcal{D}((z', u_\sigma), 1^n) = 1]| < \mathsf{negl}(n),$$

is negligible. Informally, the equalization Lemma (2.3.2) combined with Claim 3.1 yields that all variables in the first loop of PRG_Z have equal nbpe $\frac{n+\log n}{n}$. Then taking the parallel repetition in the second loop of Figure 3.1 allows us to obtain min-entropy by the flattening lemma (2.3.2). Finally, by the block source extraction Lemma (2.3.3) hashing each column with given min-entropy allows us to extract (almost) uniformly random bits. This allows us to argue that PRG_Z is a Z-seeded PRG. Details follow.

Proof of Lemma 3.1.2. Suppose $G_{nb}(x)$ is an n-block next-bit pseudoentropy generator with total next-bit pseudoentropy (nbpe) equal to $n + \log n$. In the pseudocode of PRG_Z (Fig. 3.1), it holds by Lemma 2.3.1 and Claim 3.1 that at each step $i \in [t]$ in the first loop that the blocks of each

$$y_i \leftarrow G_{nb}(x^{t+i})_{i \mod n,\ldots,n}, G_{nb}(x^t)_{<i \mod n},$$

have equal nbpe $\frac{n+\log n}{n}$, when conditioned on the hash function and previous bits.

Then, in the second loop it holds for each

$$y'_j \leftarrow (y_1[j], ..., y_t[j])$$

by directly applying Lemma 2.3.2 and parallel amplification, that each y'_j has conditional next-bit min-entropy at least

$$h_{\min} := t \cdot \frac{n + \log n}{n} - \mathcal{O}\Big((\log n + l) \cdot \log n + \sqrt{t}\Big)$$

conditioned on the hash and previous columns.

Finally, by the Block Source Extraction Lemma 2.3.3 it holds that applying hash functions with output length $t' = \lfloor h_{\min} \rfloor - \lceil \log^2 n \rceil$ it holds that we extract t' random bits with each x_j, implying we extract a total of $t'n$ bits of randomness. Thus it holds that $(h, \hat{x}_1, ..., \hat{x}_t, \hat{x})$ is computationally indistinguishable from (h, u_{tn}, u) when $|\hat{x}| = |u|$, which holds equivalently when passing through the deterministic function G_{nb}. Phrased differently, we have that

$$(h, \hat{x}_1, ..., \hat{x}_t, \hat{x}) \overset{c}{\approx} (h, u_1, ..., u_t, u),$$

which directly implies that the output of the generator PRG_Z on input z satisfies

$$\begin{aligned}\mathsf{PRG}_Z(z) &= (h, ((G_{nb}^{t+i})_{<i \mod n})_{i=1}^t, (G_{nb}(\hat{x}_i))_{i=1}^t, \hat{x}) \\ &\overset{c}{\approx} (h, ((G_{nb}^{t+i})_{<i \mod n})_{i=1}^t, (G_{nb}(u_i))_{i=1}^t, u),\end{aligned}$$

meaning PRG_Z is a Z-seeded generator.

Finally, to ensure that $|\hat{x}| > 0$ it remains to set t such that $t'n - tn > 0$, where $t' = \lfloor h_{\min} \rfloor - \lceil \log^2 n \rceil$. We actually set the difference as $\log n$ to obtain optimal stretch per invocation of the underlying function f. To do this it suffices to set t such that

$$t'n - tn > \Omega(\frac{s \log n}{n}) \iff (\lfloor h_{\min} \rfloor - \lceil \log^2 n \rceil)n - tn > \Omega(\frac{s \log n}{n}),$$

where $h_{\min} = t \cdot \frac{n+\log n}{n} - \mathcal{O}\big((\log n + l) \cdot \log n + \sqrt{t}\big)$. This is satisfied when choosing $t = \mathcal{O}(n^2 \log n)$. This then also directly implies the seed length of our Z-seeded generator, namely $s = |z| = t + 2tn = \mathcal{O}(nt) = \mathcal{O}(n^3 \log n)$. $\square$

3.1.3 Z-seeded generator to PRG

We conclude by proving Lemma 3.1.3 and show that the existence of a Z-seeded generator implies a PRG. This is acheived via a standard hybrid argument; an efficient adversary against the resulting pseudorandom generator is used to construct an adversary against our Z-seeded PRG by considering the advantage against a sequence of distributions H_i with the PRG distributions and uniform distribution as the extreme of two hybrid games.

Recall that given a Z-seeded generator G we define $G_i(z) = (G_{i-1}(z'), u'_\sigma)$ iteratively, such that $(z', u'_\sigma) = G(z)$ and G_0 is the empty string. In other words we apply G iteratively on part of its own output. For stretch it suffices to set $i = l$ such that $l\sigma > |z|$. Thus it suffices to show that G_l is pseudorandom.

Proof of Lemma 3.1.3. Let G be a Z seeded PRG on inputs from an efficiently samplable distribution Z. Suppose that for all PPT distinguishers $\mathcal{B}$ it holds that

$$|\Pr_{z \leftarrow\$ Z}[\mathcal{B}(G(z), 1^n) = 1] - \Pr_{z' \leftarrow\$ Z}[\mathcal{B}((z', u_\sigma), 1^n) = 1]| < \mathsf{negl}(n).$$

We omit the security parameter in the sequel for conciseness of notation. Suppose towards a contradiction that G^l has an efficient distinguisher $\mathcal{D}$ such that for a non-negligible function $\varepsilon(n)$ it holds that

$$|\Pr[\mathcal{D}(G_l(z)) = 1] - \Pr[\mathcal{D}(U_{l\sigma}) = 1]| = \mathbb{E}[\mathcal{D}(G_l(z)) - \mathcal{D}(U_{l\sigma})] \geq l\varepsilon.$$

Define for $i \in [l]$ hybrids H_i, such that $H_i = (G_i(z), u_{(l-i)\sigma})$, i.e.

$$\begin{aligned} H_0 &= u_{l\sigma}, \\ H_1 &= (G_1(z), u_{(l-1)\sigma}), \\ &\vdots \\ H_l &= G_l(z). \end{aligned}$$

We now define a distinguisher $\mathcal{D}_G$ against G by

$$\mathcal{D}_G(z, u) := \mathcal{D}(G_{i-1}(z), u, u_{(l-i)\sigma}).$$

where $i \in [l]$ and $u_{(l-i)\sigma}$ are sampled uniformly at random by $\mathcal{D}_G$. Note that when (z, u) equals the output of $G(z) = (z', u'_\sigma)$ it holds by definition of G_i that $(G_{i-1}(z'), u'_\sigma) = G_i(z)$, implying

$$\mathcal{D}_G(z', u'_\sigma) = \mathcal{D}(G_{i-1}(z'), u'_\sigma, u_{(l-i)\sigma}) = \mathcal{D}(G_i(z), u_{(l-i)\sigma}) = \mathcal{D}(H_i).$$

Then it holds that

$$\mathcal{D}_G(z,u) = \mathcal{D}(G_{i-1}(z), u, u_{(l-i)\sigma}) = \mathcal{D}(H_{i-1})$$
$$\mathcal{D}_G(G(z)) = \mathcal{D}_G(z', u'_\sigma) = \mathcal{D}(H_i),$$

which directly yields that the distinguishing advantage of $\mathcal{D}_G$ on (z,u) and $G(z)$ is

$$|\Pr[\mathcal{D}_G(G(z))=1] - \Pr[\mathcal{D}_G(z,u)=1]| = |\mathbb{E}[\mathcal{D}_G(G(z)) - \mathcal{D}_G(z,u)]| \quad (3.2)$$
$$= |\mathbb{E}[\mathcal{D}(H_i) - \mathcal{D}(H_{i-1})]|, \quad (3.3)$$

where the probability is over z and the internal randomness of the distinguisher $\mathcal{D}_G$. Note that i is a uniformly random integer in $[l]$. It now holds by the law of total expectation

$$\mathbb{E}[\mathcal{D}(H_i) - \mathcal{D}(H_{i-1})] = \mathbb{E}[\mathbb{E}[\mathcal{D}(H_i) - \mathcal{D}(H_{i-1}) \mid i]] \quad (3.4)$$
$$= \frac{1}{l}\sum_{k=1}^{l} \mathbb{E}[\mathcal{D}(H_k) - \mathcal{D}(H_{k-1})] \quad (3.5)$$
$$= \frac{1}{l}\,\mathbb{E}[\mathcal{D}(H_l) - \mathcal{D}(H_0)] \quad (3.6)$$
$$\geq \frac{l\varepsilon}{l} = \varepsilon \quad (3.7)$$

where (3.4) follows from the law of total expectation conditioned on i, (3.5) from the fact that i is uniform in $[l]$ and linearity of expectation. Equation (3.6) follows from cancellation by the hybrids leaving only the end terms. The final equation (3.7) then follows by the noticeable distinguishing advantage against G_l. Combined with (3.2) this contradicts the indistinguishability of (z,u) and $G(z)$. Thus the assumption that $\mathcal{D}$ has non-negligible advantage can not hold, and G_l is a PRG. □

3.2 PRG via DOWF

We now show that, in fact, the PRG construction holds when assuming only $(1-\gamma)$ distributional one-way functions, as opposed to the much stricter property of standard γ one-wayness. Recall the main steps in the PRG construction:

1 - PEG : Show that $(f(x), x_1, ..., x_n)$ is a next-bit pseudoentropy generator (PEG) with $n + \omega(\log n)$ next-bit pseudoentropy.

2 - PRG_Z: Establish that a next-bit pseudoentropy generator can be manipulated with the lemmas of Chapter 4 to arrive at a *Z-seeded PRG*.

3 - PRG: Showby a standard hybrid argument that iterating a Z-seeded PRG PRG_Z yields a PRG.

Since the PRG construction of Vadhan and Zheng is modular, we only use in Lemma 3.1.2 that f is a OWF. In fact, for any function where we can show that

$$\mathsf{KL}_{x \leftarrow\$ \{0,1\}^n}(f(x), x \| f(x), \mathcal{A}(f(x), 1^n)) > \log \frac{1}{\gamma}$$

the proof of a next-bit pseudoentropy gnerator to Z-seeded PRG to PRG follows identically. It thus suffices to establish the following theorem to obtain a tight construction of a PRG from a distributional OWF.

Theorem 3.2.1 (Distributional one-way functions have Nbpe)**.** *Let f be a $(1-\gamma)$ distributional one-way function. Then $G_{nb}(x) = (f(x), x)$ has next-block pseudoentropy at least $n + \log \frac{1}{\gamma} - \log(2-\gamma)$. In particular if f is a strong distributional one-way function, the generator G_{nb} has conditional next-block pseudoentropy at least $n + \omega(\log n) - 1$.*

Note that setting $\gamma = \mathsf{negl}(n)$ we thus obtain a generator asymptotically equal to the one obtained via a strong one-way function in [37].

The proof of Theorem 3.2.1 follows directly from the following lemma.

Lemma 3.2.2 (Distributional one-way functions have relative pseudoentropy)**.** *Let f be $(1-\gamma)$ distributionally one-way. Then it holds that*

$$\mathsf{KL}_{x \leftarrow\$ \{0,1\}^n}(f(x), x \| f(x), \mathcal{A}(f(x), 1^n)) > \log \frac{1}{\gamma} - \log(2-\gamma).$$

Proof of Lemma 3.2.2. Let f be a $(1-\gamma)$ distributional one-way function. The proof follows from applying Bretgnolle-Huber's inequality (1.1.3), with setting the distributions as $p = (f(x), x)$, and $q = (f(x), \mathcal{A}(f(x))$. By definition it holds for f that $\mathsf{SD}(p, q) > 1 - \gamma$, and thus

$$\mathsf{KL}_{x \leftarrow\$ \{0,1\}^n}((f(x), x) \| (f(x), \mathcal{A}(f(x)))) > \log \frac{1}{1-(1-\gamma)^2} = \log \frac{1}{\gamma} - \log(2-\gamma).$$

□

In particular if a function f is a strong distributional one-way function it has $\omega(\log n) - 1$ relative pseudoentropy. This directly yields an identical PRG construction as from standard OWFs by replacing Lemma 3.1.2 with the above distriutional version. This result is due to the fact that as the

notion of next-block pseudoentropy is equivalent to relative entropy for short blocks, meaning that a bound on statistical distance can be used to obtain a bound on pseudoentropy directly. We will discuss the relationship between distributional primitives and computational entropy-based constructions further in Chapter 6.

Chapter 4

Statistically hiding commitments from distributional collision resistance

In this chapter our central goal is to tighten the construction of a statistically hiding commitment scheme (SHC) from a distributionally collision resistant hash function (dCRHF) that is presented by Bitansky, Haitner, Komargodski and Yegev [7], which uses the construction of SHCs from inaccessible entropy generators due to Haitner, Reingiold, Vadhan and Wee [19]. Recall that for a hash function to be distributionally collision resistant it should be hard to sample *uniform* collisions, i.e. it holds that

$$\mathsf{SD}((h, \mathcal{A}(h, 1^n), (h, \mathrm{Col}(h)) = \mathbb{E}_{h \leftarrow\$ \mathcal{H}}[\mathsf{SD}(\mathcal{A}(h, 1^n), \mathsf{Col}(h)] > \frac{1}{p},$$

for some positive polynomial $p > 1$. The core argument in the proof of the original construction, Lemma 4 of the original paper, establishes that given a dCRHF $h \leftarrow\$ \mathcal{H}$ and a uniformly random x, the function $(h(x), x)$ is a two-block *inaccessible entropy generator* (Def. 1.1.9), which directly implies the construction of a constant round SHC by the results of HRVW. We improve the core Lemma via applying the tighter Bretagnolle-Huber bound 1.1.3, instead of Pinsker's inequality as in [7] to obtain an upper bound on the statistical distance in the definition of h.

Compared to the accessible entropy of at most $n - \frac{(p-1)^2}{4p^2}$ from a $\left(1 - \frac{1}{p}\right)$-dCRHF, where $p > 1$ is a positive polynomial, achieved by Bitansky, Haitner, Komargodski and Yegev, we obtain that the accessible entropy is at most $n - \log\left(\sqrt{\frac{p^2}{2p-1}}\right)$. For concreteness, when the statistical distance in the definition of the dCRHF h is close to 1 (i.e. h is "hard"), then the original bound tends to $\frac{1}{4}$, while our bound tends to roughly $\log(p)$ (ignoring constants).

In Section 4.1 we review the construction of SHCs from an inaccessible entropy generator. Although we use the original result in a black box way, we give an outline of the construction to contextualize the concrete improvements obtainable by a tighter dCRHF accessible entropy bound. Afterwards we present the new bound on inaccessible entropy from a dCRHF in Section 4.2 and analyze the resulting commitment scheme in Section 4.3. We note that a precise analysis of the complexity of the SHC from dCRHF is not included in the original BHKY construction, meaning we can't simply use their results in a black-box way. Thus we also need the HRVW construction of commitments from accessible entropy to understand the effect of an improved bound.

4.1 SHC from inaccessible entropy generator

4.1.1 SHC from an arbitrary block generator

For completeness, we provide a self-contained overview of the core arguments in the analysis of the construction of a SHC from and inaccessible entropy generator presented by HRVW [19], which is almost analogous in structure to the PRG construction presented in Chapter 3. The outline of the construction is roughly as follows: Let G be an m-block inaccessible entropy generator (i.e. G has real entropy $k(n)$ and accessible entropy at most $k_{acc} = k(n) - \delta(n)$).

(1). Equalize: Given an inaccessible entropy generator G, we equalize the real and accessible entropy by taking several independent copies and concatenating the outputs. By Lemma 2.3.1 this yields accessible, real

(2). Flatten: Then, parallel repetition flattens the equalized blockwise entropy bounds to min-entropy.

(3). Hash in parallel: Finally, we obtain a weakly binding commitment scheme Com via a hashing protocol, which when run with $t = \frac{\log(n)^2 m}{\delta}$ independent parallel instance of the weakly yields a statistically hiding and computationally binding commitment.

Note that the only real differences compared to the Z-seeded PRG construction of Chapter 3 are the different notions of entropy in the analysis, and the different hashing methods in the last step. This is why the third step is presented as a single step rather than two smaller steps — we wish to underline the fact that constructions from PRGs and SHCs are fundamentally very similar. Indeed the operations performed are equivalent, with a final hashing

based argument as the last step. The full proof of the construction along with analysis is contained in [19], with a simpler version being available in [20]. We present the pseudocode for the weakly binding commitment scheme Com in Figure 4.1, and then review the construction in the special case when G is a *constant block generator* (i.e. the number of blocks m does not grow with the security parameter n), as is the case when working with a dCRHF based generator.

We define an m-block generator as follows:

Definition 4.1.1 (m-block generator)**.** *We say a polynomial-time computable (deterministic) function $G : \{0,1\}^{p(n)} \times \{0,1\}^{s(n)} \to (\{0,1\}^{l(n)})^{m(n)}$ is an m-block generator with blocks of length l on the public parameter p and seed s.*

Note that the functions determining the input and output lengths of the generator G are assumed to be functions of the underlying security parameter n. We will be particularly interested in the entropy thresholds $k := k(n)$ of the individual blocks of G. We stress that all functions are in terms of n.

Let G be a an m-block generator, where *each block* of G has real min-entropy at least $k(n) \geq 3n$, and the total inaccessible entropy of G is at least $4mn$. Suppose that $\mathcal{H}^1, \mathcal{H}^2$ are efficient l-wise and 2-wise independent hash function families respectively (see Definition 2.1.4), with $\mathcal{H}^1$ mapping strings of length n to strings of length $k - 3n$ and $\mathcal{H}^2$ mappings strings of length n to n. Let $\mathcal{F}$ be a family of universal one-way hash functions (see Definition 2.1.6) mapping strings of length n to strings of length n.[1] All three families are assumed samplable and public in the subsequent protocol. Furthermore, recall that variables distributed according to the Bernoulli distribution $\mathrm{Ber}(p)$ are 0-1 variables taking the value 1 with probability p.

We define the weakly binding commitment scheme (see the discussion after Def. 2.1.3) between parties S and R using standard protocol syntax. All computation on either party's side is done in private with the other party only gaining information via communication denoted by the message arrows. Recall that $y[i]$ denotes the i-th block of the random variable y. In other words, the receiver samples a public parameter p and sends it to S. The sender then computes in private a random execution of G using this public parameter and a secret random value x. Looping through the blocks, the receiver requests at round i either the i-th block of the generator output, or the i-th block processed through a hashing protocol. In the subsequent case the sender then masks its input bit via XORing it with $y[i]$ by extracting

[1] All three families are efficiently constructable and efficiently samplable as shown by Carter and Wegman, and Katz and Koo.

$\mathsf{Com}(1^n)$

Sender S		**Receiver** R
`private:` $b \in \{0,1\}$		
	$\xleftarrow{\quad p \quad}$	$p \leftarrow\!\$\, \{0,1\}^c$
$x \leftarrow\!\$\, \{0,1\}^s$		
$y \leftarrow G(p,x)$		
..... for $i = 1$ to m		
	$\xleftarrow{\quad c_i \quad}$	$c_i \leftarrow\!\$\, \mathrm{Ber}(\frac{1}{m+1-i})$
if $c_i == 0$	$\xrightarrow{\quad y[i] \quad}$	next loop
..... **else**		
	$\xleftarrow{\quad h^1 \quad}$	$h_1 \leftarrow\!\$\, \mathcal{H}^1$
$y^1 \leftarrow h^1(y[i])$	$\xrightarrow{\quad y^1 \quad}$	
	$\xleftarrow{\quad h^2 \quad}$	$h_2 \leftarrow\!\$\, \mathcal{H}^2$
$y^2 \leftarrow h^2(y[i])$	$\xrightarrow{\quad y^2 \quad}$	
	$\xleftarrow{\quad f \quad}$	$f \leftarrow\!\$\, \mathcal{F}$
$y^f \leftarrow f(y[i])$	$\xrightarrow{\quad y^f \quad}$	
$u \leftarrow\!\$\, \{0,1\}^l$		
	$\xrightarrow{(\langle u, y[i]\rangle \oplus b, u)}$	end execution

Figure 4.1: *Weakly* binding commitment Com

a weakly random bit from it by taking the inner product with a random, public string u. After this the parties end the execution. Then taking $t = \frac{\log(n)^2 m}{\delta}$ parallel and independent executions of Com yields a SHC consistent with Definition 2.1.3, as shown in the proof of Lemma 6.5 in HRVW [19]. The intuition behind the protocol is that, when conditioned on the previous blocks, after the hashing interaction $y[i]$ has high real min-entropy, but "low" accessible entropy.

In the opening phase of the resulting commitment scheme the sender simply reveals the private bit b and its sampled x, and the receiver checks whether this yields the same as the final calculation

$$\langle u, y[i] \rangle \oplus b, u),$$

as well as checking the generated blocks $y[i]$. Intuitively, hiding is due to the real min entropy of each block $y[i]$ being "high" when conditioned on the previous transcript, whereas the binding is due to an upper bound on the accessible entropy implying that $y[i]$ is fixed (from a computational perspective) for all bounded senders. Specifically, any computationally bounded sender can not cheat by picking a new secret x and computing a valid transcript $y[1], ..., y[i]$, as any bounded algorithm can only access a small subset of the support of G.

We include a visualization for the concrete SHC from the weakly binding commitment in Fig 4.2 below:

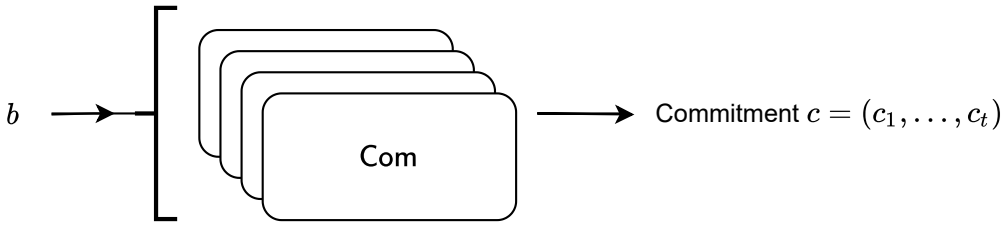

Figure 4.2: SHC from weakly binding commitment

4.1.2 SHC from a constant block generator

As we will later focus use the two-block generator $(h(x), x)$ for the construction, we can further improve on the previous commitment scheme construction. Namely, when the generator has a *constant* number of blocks (i.e. when

then input grows with n the number if blocks stays the same) we obtain an almost identical commitment scheme construction, with a few tweaks.

The largest difference compared to the general case is that we can skip the entropy equalization step (Step 1). Centrally, knowing the number of blocks is constant allows us to directly use parallel repetition and Lemma 2.3.2. The problem then becomes that the per-block entropy thresholds are not known in general, and not equal as in the "standard" construction. Since we have constant blocks however, we can instead use a variant of Com for all possible values for the blockwise entropies (up to accuracy determined by the inaccessible entropy) independently in parallel. This ends up saving in the round complexity however, as only a constant number of rounds is required. HRVW show that one may thus obtain polynomially many commitments where all are binding and *at least 1* is hiding, yielding a SHC when combining all commitments by taking the bitwise XOR.

We define the constant block weakly binding commitment as $\widetilde{\mathsf{Com}}(\tilde{k}, 1^n)$ with the following differences compared to Com:

- $\widetilde{\mathsf{Com}}$ takes as input an advice string $\tilde{k} = (k_1, k_2, ..., k_m)$ denoting a "guess" for the blockwise min-entropies of G.
- Instead of using the hash family $\mathcal{H}^1$ from n to $k - 3n$ bits in each round of the loop, $\widetilde{\mathsf{Com}}$ instead uses separate $\mathcal{H}_i$ mapping strings of length n to $k_i - 3n$ bits (if $k_i = 0$ $\widetilde{\mathsf{Com}}$ simply skips the round).

That is to say $\widetilde{\mathsf{Com}}$ uses the advice string it gets as input to choose the correct hash function at each round of its execution. This process is repeated for each possible tuple $\tilde{k} = (k_1, k_2, ..., k_m)$, where

1. $k_i \in \{0, \frac{\delta}{2m}, ..., \lfloor l \rfloor_{\delta/2m}\}$, since the maximal entropy of each block is the total length, and

2. $\sum k_i \in [(1 - \delta/2) \cdot k, k]$, as the total entropy has to be k (up to additive error).

We say such tuples are (δ, m)-*valid*. Other than these two details the constructions (and indeed the proofs) of the both protocols are identical. Note that if the chosen tuple $\tilde{k} = (k_1, k_2, ..., k_m)$ corresponds with the real blockwise entropies, then the resulting protocol is equivalent to the standard case up to some polynomial loss due to the accuracy.

Thus, to construct a SHC from a constant block inaccessible entropy generator, we take the parallel repetition of G to obtain min-entropy and amplify the weakly binding commitment $\widetilde{\mathsf{Com}}$ in parallel for each possible value of blockwise min-entropies $\tilde{k}$. The number of necessary parallel instances to

construct a SHC from a generator with a constant number of blocks is determined entirely by the number of all (δ, m)-valid tuples $\tilde{k}$, which we denote by K_n. As the accuracy δ yields a smaller number of (δ, m)-valid tuples (as the number of possibilities in the first constraint becomes smaller), an improved bound on inaccessible entropy of G will yield improved complexity in terms of parallel instances. We analyze the exact improvement in Section 4.3 after establishing an improved bound compared to that of BHKY on inaccessible entropy of $G(h, x) = (h(x), x)$ in Section 4.2.

4.2 Accessible entropy generator from dCRHF

In this chapter we present our improvement on the original construction of a SHC from distributional collision resistance due to BHKY. Recall that we are interested in establishing a tighter bound on the accessible entropy of the generator $(h(x), x)$, where h is a dCRHF, which will further improve the resulting SHC construction. We define a simple two-block generator using any dCRHF h as follows:

Definition 4.2.1. *Let the function* $G : \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^m \times \{0,1\}^n$ *be defined as*

$$G(h, x) = (h(x), x).$$

Bitansky, Haitner, Komodorski and Yegev show that G for a random h as public parameter is a two block inaccessible entropy generator with non-negligible inaccessible entropy. We slightly reword Lemma 4 from the original paper as the following theorem:

Theorem 4.2.1. *There exists a non-negligible function* $\delta(n)$ *such that for any* G*-consistent generator* $\widetilde{G}$ *as defined in 1.1.9 conditioned on its internal coin tosses* R_1, R_2*, it holds that*

$$\sum_{i=1}^{m} \mathrm{H}\Big(\widetilde{G}[i] \;\Big|\; S, R_{<i}\Big) = \mathrm{H}\Big(\widetilde{G}[1]\Big) + \mathrm{H}\Big(\widetilde{G}[2] \;\Big|\; R_1\Big) \leq n - \delta(n),$$

i.e. the accessible entropy of G *is at most* $n - \delta(n)$*.*

Specifically BHKY show by using Pinsker's inequality that for a $1/p$ dCRHF choosing $\delta = \frac{1}{4p^2}$ and assuming the contrapositive to Theorem 6.2.1 yields to the inequality

$$\mathsf{SD}((h, \mathcal{A}(h, 1^n), (h, \mathsf{Col}(h)) < 1/p,$$

contradicting the definition of the dCRHF h, where $\mathcal{A}$ and Col are as in Def. 1.1.12. This in turn implies that G is an inaccessible entropy generator with

$\frac{1}{\mathsf{poly}(n)}$ inaccessible entropy. We present a slightly tighter bound on δ that is approximately $\log(p)$ for any $1 - 1/p$ dCRHF (compared to the at most constant bound of the original result).

Corollary 4.2.1.1. *Suppose h is a $1 - \frac{1}{p}$ dCRHF. Then we may set $\delta = \log\left(\sqrt{\frac{p^2}{2p-1}}\right)$ as the function in Theorem 6.2.1.*

We prove Theorem 4.2.1 following the line of reasoning outlined by BHKY, with the main difference being using Bretagnolle-Huber instead of Pinsker's inequality to relate the statistical distance and relative entropy. Then Jensen's inequality together with bounds on the relative entropies shown by BHKY yield the result.

Proof. Let $\mathcal{H}$ be a $(1 - 1/p)$ dCRHF family and $\mathcal{H}$ be a collision resistant hash family. Suppose towards a contradiction that there exists some G-consistent generator $\widetilde{G}$ using public parameter P and internal randomness $R = (R_1, ..., R_m)$ that contradicts the upper bound on the accessible entropy of $G(h, x) = (h(x), x)$ for infinitely many n, i.e.

$$\sum_{i=1}^{m} \mathrm{H}\left(\widetilde{G}[i] \;\middle|\; P, R_{<i}\right) > n - \delta.$$

We denote the number of bits needed to sample internal randomness R_i by $\{0,1\}^s$. We use $\widetilde{G}$ to break the security of $\mathcal{H}$ as defined in Def. 1.1.12 as follows:

$\mathcal{A}(h, 1^n)$

$r \leftarrow\!\!\$\, \{0,1\}^s$
$y \leftarrow \widetilde{G}(h, r)$
$r_1 \leftarrow\!\!\$\, \{0,1\}^s$
$r_2 \leftarrow\!\!\$\, \{0,1\}^s$
$x_1 \leftarrow \widetilde{G}(h, r, r_1)$
$x_2 \leftarrow \widetilde{G}(h, r, r_2)$
return (x_1, x_2)

In other words, given a hash function h as input the adversary runs $\widetilde{G}$ to obtain a hash value y (corresponding to the first block), then runs $\widetilde{G}$ twice to obtain two strings x_1, x_2 that are mapped to y (two separate instances of the second block). We claim that due to the consistency of $\widetilde{G}$ (i.e. the outputs of the emulator are always in the support of G) the adversary succeeds in

sampling close to uniformly from the pre-image distribution $h^{-1}(y)$ if the accessible entropy is too large.

We denote by x_1, x_2 the outputs of the adversary, and by c_1, c_2 the samples from the ideal collision distribution (Def. 1.1.11). By definition of the adversary and the ideal collision distribution it holds that

$$\mathsf{SD}((x_1, x_2), (c_1, c_2)) = \mathsf{SD}(\mathcal{A}(h, 1^n), \mathsf{Col}(h).$$

Taking expectations and conditioning on the randomness of h allows us to write

$$\mathbb{E}_{h \leftarrow\$ \mathcal{H}}[\mathsf{SD}(\mathcal{A}(h, 1^n), \mathsf{Col}(h)] = \mathsf{SD}((h, \mathcal{A}(h, 1^n), (h, \mathsf{Col}(h)),$$

meaning that the expectation $\mathbb{E}[\mathsf{SD}((x_1, x_2), (c_1, c_2))]$ taken over the hash family is exactly the statistical distance in the definition of the distributional collision resistance.

By the Bretagnolle-Huber inequality 1.1.3 it holds that

$$\begin{aligned}\mathsf{SD}((x_1, x_2), (c_1, c_2)) &\leq \sqrt{1 - 2^{-\mathsf{KL}((x_1,x_2)||(c_1,c_2))}} \\ &= \sqrt{1 - 2^{-\mathsf{KL}(x_1||c_1) - \mathbb{E}\left[\mathsf{KL}(x_2||c_2)_{|x_1}\right]}},\end{aligned}$$

where the last inequality is due to the chain rule of relative entropy (which follows from the chain rule of Shannon entropy in Lemma 1.1.1 and conditioning on x_1). Taking expectations over $\mathcal{H}$ and the internal randomness of the adversary then yields

$$\begin{aligned}\mathbb{E}[\mathsf{SD}((x_1, x_2), (c_1, c_2))] &\leq \mathbb{E}\left[\sqrt{1 - 2^{-\mathsf{KL}(x_1||c_1) - \mathbb{E}\left[\mathsf{KL}(x_2||c_2)_{|x_1}\right]}}\right] \\ &\leq \sqrt{\mathbb{E}\left[1 - 2^{-\mathsf{KL}(x_1||c_1) - \mathbb{E}\left[\mathsf{KL}(x_2||c_2)_{|x_1}\right]}\right]} \qquad (4.1) \\ &= \sqrt{\mathbb{E}[1] - \mathbb{E}\left[2^{-\mathsf{KL}(x_1||c_1) - \mathbb{E}\left[\mathsf{KL}(x_2||c_2)_{|x_1}\right]}\right]} \\ &\leq \sqrt{1 - 2^{-\mathbb{E}[\mathsf{KL}(x_1||c_1)] - \mathbb{E}\left[\mathsf{KL}(x_2||c_2)_{|x_1}\right]}} \qquad (4.2) \\ &< \sqrt{1 - 2^{-2\delta}} \qquad (4.3)\end{aligned}$$

where (4.1) follows from Jensen's inequality and (4.2) follows from applying Jensen's inequality on the exponential function and the expectation of a constant (recall that square root is concave, and the exponential function is convex thus in turn the negative exponential function is concave). The inaccessible entropy bound in (4.3) follows from the bounds derived on both

relative entropy terms by BHKY (Claim 1 and Claim 2 in the original paper [7], namely if the accessible entropy is greater than $n - \delta$ then it holds that

$$\mathbb{E}[\mathsf{KL}(x_1||c_1)] < \delta, \text{ and}$$
$$\mathbb{E}\left[\mathsf{KL}(x_2||c_2)_{|x_1}\right] < \delta.$$

Although a multiplicative constant $\frac{1}{4}$ seems to be missing from the claims in the final inequalities of the paper (as they conclude $\leq 1/q = 1/p^2$, despite setting $q = 4p^2$), the actual derived inequalities in terms of the inaccessible entropy do yield the above bounds. Setting $\delta = \log\left(\sqrt{\frac{p^2}{2p-1}}\right)$ in (4.3) allows us to conclude

$$\begin{aligned}\mathbb{E}[\mathsf{SD}((x_1, x_2), (c_1, c_2))] &< \sqrt{1 - \frac{2p-1}{p^2}} \\ &= \sqrt{\frac{p^2 - 2p + 1}{p^2}} \\ &= \sqrt{\frac{(p-1)^2}{p^2}} = 1 - 1/p\end{aligned}$$

which is a contradiction against the statistical distance of h. □

It therefore follows that the generator G has accessible entropy at most $n - \log\left(\sqrt{\frac{p^2}{2p-1}}\right)$ when h is $(1 - 1/p)$ distributionally collision resistant. The quantitative difference in entropy can be observed from figure 4.3 below, with the bounding function $2\sqrt{\delta}$ by BHKY in blue, and our bounding function $\sqrt{1 - 2^{-2\delta}}$ in red. We observe that in order to contradict some $(1-1/p)$ lower

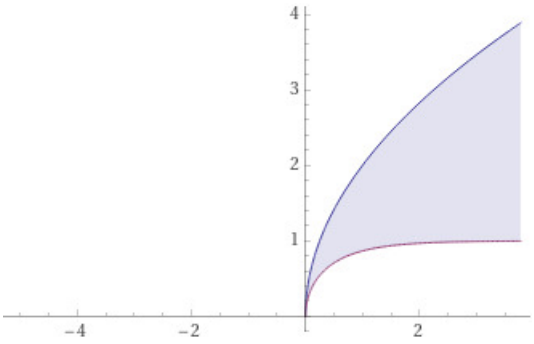

Figure 4.3: Accessible entropy bounds

bound on the statistical distance, the function $\sqrt{1 - 2^{-2\delta}}$ necessitates larger δ, whereas to obtain a contradiction $2\sqrt{\delta} \leq 1 - 1/p$ it holds that $\delta \in (0, \frac{1}{4})$.

4.3 SHC from distributional collision resistance

Combining the results of the two previous sections allows us to analyze the number of tuples in the set of all advice strings K_n needed in the constant block SHC construction of Section 4.1. Let G be as in Section 4.2 and $\mathcal{H}$ be a $(1-1/p)$ dCRHF family. Note that the first block of G is of length $l = |h(x)|$ by definition. Recall that K_n denotes the set of possible (δ, m)-valid tuples $\widehat{k} = (k_1, ..., k_m)$ of candidate entropies for each block of the underlying inaccessible entropy generator with real entropy k and accessible entropy at most $k - \delta$, where it holds that

$$k_i \in \left\{0, \frac{\delta}{2m}, ..., \lfloor l \rfloor_{\delta/2m}\right\}, \text{ and}$$
$$\sum_{i\in[m]} k_i \in [(1-\delta/2)\cdot k, k].$$

As G only consists of two blocks, choosing $k_1 \in \left\{0, \frac{\delta}{2m}, ..., \lfloor l \rfloor_{\delta/2m}\right\}$ fixes k_2, due to the fact that

$$\sum_{i\in[m]} k_i = k_1 + k_2 \in [(1-\delta/2)\cdot k, k].$$

In other words since the total entropies sum to k (with additive error), choosing the value of k_1 fixes the value of k_2. The number of tuples in K_n is then determined completely by the total number of possible values of k_1, i.e. $\lfloor l(n) \rfloor_{\delta/2m} = \lfloor l(n) \cdot \frac{2m}{\delta} \rfloor$, where l is the output length of the hash function h. Setting $m = 2$ and $\delta = \log\left(\sqrt{\frac{p^2}{2p-1}}\right)$ by Corollary 4.2.1.1 yields

$$|K_n| = \lfloor l(n) \cdot \frac{2m}{\delta} \rfloor = \left\lfloor \frac{4l}{\log\sqrt{\frac{p^2}{2p-1}}} \right\rfloor,$$

which we will denote

$$|K_n| =_{\lfloor\cdot\rfloor} \frac{4l}{\log\sqrt{\frac{p^2}{2p-1}}}.$$

The same analysis applied to the polynomial bound $\delta = \frac{(p-1)^2}{4p^2}$ of BHKY (which is obtained when considering a $(1-1/p)$ dCRHF by using their methods) we instead obtain $|K_n| =_{\lfloor\cdot\rfloor} \frac{16lp^2}{(p-1)^2}$, which is asymptotically greater than the logarithmic bound $\delta = \log\left(\sqrt{\frac{p^2}{2p-1}}\right)$.

We may note that relatively simple calculus suffices to show that the inequality

$$\frac{16lp^2}{(p-1)^2} > \frac{4l}{\log\sqrt{\frac{p^2}{2p-1}}}$$

always holds when $p > 1$. As polynomials smaller than 1 yield a contradictory bound on the statistical distance in the definition of dCRHF it is already an implicit assumption that $p > 1$. More specifically if we have a $1/p$ dCRHF setting $p < 1$ yields statistical distance greater than 1, and if we have a $(1 - 1/p)$ dCRHF setting $p < 1$ yields negative statistical distance. Ignoring the $4l$ factor common in both expressions we obtain the following graph:

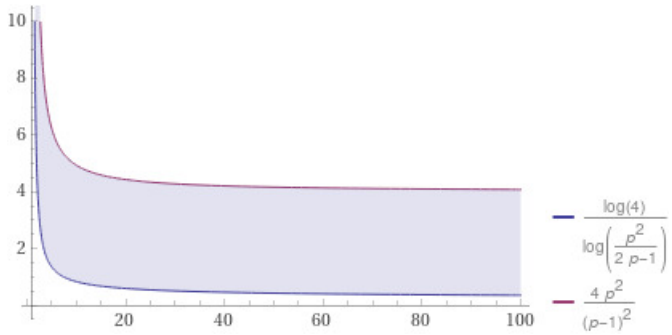

Figure 4.4: Parallel instances in SHC construction

where we observe that the logarithmic bound of $\frac{1}{\log\sqrt{\frac{p^2}{2p-1}}}$ yields complexity asymptotically tending to zero, whereas the polynomial bound $\frac{4p^2}{(p-1)^2}$ tends to 4. This is simple to confirm by taking limits. Note that the $4l$ factor should in reality not be ignored, so "realistic" parameter choices should still yield large (and not close to 0) number of parallel instances. Choosing e.g. $l = 256, p = 10$ (i.e. the statistical distance against achievable against a dCRHF with output length $l = 256$ is at least $1 - 1/p = 0.9$) yields parallel rounds numbering 427 with the logarithmic bound and 2904 for the polynomial bound.

Chapter 5

Open problems and implications

Accessible entropy from distributional one-way functions The previously presented results in Chapter 3 and 4 establish that distributional one-wayness implies next-block pseudoentropy, and that distributional *collision resistance* implies accessible entropy. An immeidate question that arises is whether accissible entropy can *also* be shown to exist based on distributional one-wayness rather than distributional collision resistance. As pointed out by Agrawal, Chen, Horel and Vadhan [1] the notions of accessible entropy and pseudoentropy seem "dual" in current constructions. Namely, the constructions of SHCs from accessible entropy and PRGs from pseudoentropy have *almost* identical structure: having some bound ($n + \log n$ and $n - \log n$ respectively) on the computational entropies, then by equalization, flattening and some kind of hashing. ACHV further show that via a notion they define as *hardness in relative entropy* it is indeed possible to get $\log n$ in both pseudo- and inaccessible entropy from (standard) OWFs, providing some further justification for the connection.

Additionally, as observed in the article constructing universal one-way hash function based on accessible entropy by Haitner, Holenstein, Reingold, Vadhan and Wee [17], a distributional one-way function suffices for the construction of a universal one-way hash function. It is an interesting question whether the construction of statistically hiding commitments based on accessible entropy can also be shown to hold nearly equivalently for strong distributional one-way functions. The approach by either ACHV [1] or HRVW [19] for establishing inaccessible entropy of $(f(x), x)$ might be possible to relax via similar statistical distance bounds as for the PRG case to apply for distributional one-way functions. This would again "unify" the notions of computational entropy that we see in the existing paradigm, but for distributional OWFs. If even accessible entropy from distributional OWFs were possible to obtain, then all three primitives based on inaccessible entropy and

pseudoentropy (PRGs, SHCs, UOWHFs) would hold *equivalently* to their standard OWF versions. If not, then we have a property pointing to a difference in the notions of computational entropy. Another option, which might be more reasonable to expect, is that standard one-way functions admit to notably more efficient black-box constructions compared to distributional one-way functions. A possible stumbling block may be that average case notions of entropy can not capture this expected efficiency improvement as they only model the hardness *on the average* taken over the entire distribution, hiding the worst-case pathology of distributional one-way functions.

Similarity and separation results Another interesting direction w.r.t. distributional OWFs is establishing exact results in constructions from distributional OWFs to standard OWFs, and tight bounds on how known OWF results translate to the distributional setting. It is perhaps reasonable to conjecture that *amplification* of a somewhat hard function to a hard function that is possible for OWFs also holds distributionally. An example would be showing that Yao's seminal result establishing that concatenating a $\frac{1}{p}$-weakly OWF polynomially many times yields a strong OWF, should also work similarly for distributional OWFs. Here a large issue in the reduction is (seemingly) that an adversary can not in general know which pre-images are easy and which are hard, causing issues for the standard repetition argument.

Furthermore, it is perhaps interesting to note that the construction of PRGs due to Vadhan and Zheng using a distributional OWF is seemingly more efficient than the current way of obtaining a strong one-way function from a strong distributional one-way function by going from a distributional OWF to a weak OWF using Impagliazzo and Luby [22], and from weak to strong by Yao's amplification results. Notably both transformations proceed in a remarkably similar way, in that they take independent repetitions, random truncations and perform random hashing. How tightly do the notions of distributional one-wayness and one-wayness relate, and is it possible to improve on the Vadhan-Zheng result for amplification?

Finally there exist applications where distributional OWFs will not admit to equivalent constructions. For a simple example, Lamport's signature scheme using OWFs [26] will not remain secure when using a distributional OWF. Understanding the exact relation between distributional OWFs and OWFs would perhaps also allow for further separations, and allow for better understanding of which types of problems can yield useful computational hardness.

Bibliography

[1] Agrawal, R., Chen, Y.-H., Horel, T., and Vadhan, S. P. Unifying computational entropies via kullback-leibler divergence. In *CRYPTO 2019, Part II* (Aug. 2019), A. Boldyreva and D. Micciancio, Eds., vol. 11693 of *LNCS*, Springer, Heidelberg, pp. 831–858.

[2] Applebaum, B., Ishai, Y., and Kushilevitz, E. Cryptography with constant input locality. In *CRYPTO 2007* (Aug. 2007), A. Menezes, Ed., vol. 4622 of *LNCS*, Springer, Heidelberg, pp. 92–110.

[3] Barak, B. The complexity of public-key cryptography. Cryptology ePrint Archive, Report 2017/365, 2017. `https://ia.cr/2017/365`.

[4] Beimel, A., Ishai, Y., Kushilevitz, E., and Malkin, T. One-way functions are essential for single-server private information retrieval. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA* (1999), J. S. Vitter, L. L. Larmore, and F. T. Leighton, Eds., ACM, pp. 89–98.

[5] Berman, I., Degwekar, A., Rothblum, R. D., and Vasudevan, P. N. Statistical difference beyond the polarizing regime. In *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part II* (2019), D. Hofheinz and A. Rosen, Eds., vol. 11892 of *Lecture Notes in Computer Science*, Springer, pp. 311–332.

[6] Berman, I., Haitner, I., and Tentes, A. Coin flipping of *any* constant bias implies one-way functions. In *46th ACM STOC* (May / June 2014), D. B. Shmoys, Ed., ACM Press, pp. 398–407.

[7] Bitansky, N., Haitner, I., Komargodski, I., and Yogev, E. Distributional collision resistance beyond one-way functions. In *EUROCRYPT 2019, Part III* (May 2019), Y. Ishai and V. Rijmen, Eds., vol. 11478 of *LNCS*, Springer, Heidelberg, pp. 667–695.

[8] Bretagnolle, J., and Huber, C. Estimation des densités : risque minimax. *Séminaire de probabilités de Strasbourg 12* (1978), 342–363.

[9] Chen, Y., Göös, M., Vadhan, S. P., and Zhang, J. A tight lower bound for entropy flattening. In *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA* (2018), R. A. Servedio, Ed., vol. 102 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 23:1–23:28.

[10] Cover, T. M., and Thomas, J. A. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, USA, 2006.

[11] Dodis, Y., Ostrovsky, R., Reyzin, L., and Smith, A. D. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput. 38*, 1 (2008), 97–139.

[12] Dubrov, B., and Ishai, Y. On the randomness complexity of efficient sampling. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006* (2006), J. M. Kleinberg, Ed., ACM, pp. 711–720.

[13] Gao, Z., Han, Y., Ren, Z., and Zhou, Z. Batched multi-armed bandits problem, 2019.

[14] Gennaro, R., Gertner, Y., Katz, J., and Trevisan, L. Bounds on the efficiency of generic cryptographic constructions. *SIAM J. Comput. 35*, 1 (2005), 217–246.

[15] Goldreich, O. A note on computational indistinguishability. *Inf. Process. Lett. 34*, 6 (1990), 277–281.

[16] Goldreich, O., Goldwasser, S., and Micali, S. On the cryptographic applications of random functions. In *CRYPTO'84* (Aug. 1984), G. R. Blakley and D. Chaum, Eds., vol. 196 of *LNCS*, Springer, Heidelberg, pp. 276–288.

[17] Haitner, I., Holenstein, T., Reingold, O., Vadhan, S. P., and Wee, H. Universal one-way hash functions via inaccessible entropy. In *EUROCRYPT 2010* (May / June 2010), H. Gilbert, Ed., vol. 6110 of *LNCS*, Springer, Heidelberg, pp. 616–637.

[18] Haitner, I., Reingold, O., and Vadhan, S. P. Efficiency improvements in constructing pseudorandom generators from one-way functions.

In *42nd ACM STOC* (June 2010), L. J. Schulman, Ed., ACM Press, pp. 437–446.

[19] Haitner, I., Reingold, O., Vadhan, S. P., and Wee, H. Inaccessible entropy I: inaccessible entropy generators and statistically hiding commitments from one-way functions. *CoRR abs/2010.05586* (2020). `https://arxiv.org/abs/2010.05586` A preliminary version appeared in STOC 2009.

[20] Haitner, I., and Vadhan, S. *The Many Entropies in One-Way Functions*. Springer International Publishing, Cham, 2017, pp. 159–217.

[21] Håstad, J., Impagliazzo, R., Levin, L. A., and Luby, M. A pseudorandom generator from any one-way function. *SIAM Journal on Computing 28* (1999), 12–24.

[22] Impagliazzo, R., and Luby, M. One-way functions are essential for complexity based cryptography (extended abstract). In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989* (1989), IEEE Computer Society, pp. 230–235.

[23] Impagliazzo, R., and Luby, M. One-way functions are essential for complexity based cryptography (extended abstract). In *30th FOCS* (Oct. / Nov. 1989), IEEE Computer Society Press, pp. 230–235.

[24] Katz, J., and Koo, C.-Y. On expected constant-round protocols for byzantine agreement. In *CRYPTO 2006* (Aug. 2006), C. Dwork, Ed., vol. 4117 of *LNCS*, Springer, Heidelberg, pp. 445–462.

[25] Komargodski, I., and Yogev, E. On distributional collision resistant hashing. In *CRYPTO 2018, Part II* (Aug. 2018), H. Shacham and A. Boldyreva, Eds., vol. 10992 of *LNCS*, Springer, Heidelberg, pp. 303–327.

[26] Lamport, L. Constructing digital signatures from a one way function. Tech. rep., October 1979.

[27] Naor, M., and Rothblum, G. N. Learning to impersonate. In *Proceedings of the 23rd International Conference on Machine Learning* (New York, NY, USA, 2006), ICML '06, Association for Computing Machinery, p. 649–656.

[28] Ostrovsky, R. One-way functions, hard on average problems, and statistical zero-knowledge proofs. In *Proceedings of the Sixth Annual Structure in Complexity Theory Conference, Chicago, Illinois, USA, June 30 - July 3, 1991* (1991), IEEE Computer Society, pp. 133–138.

[29] Ostrovsky, R., and Wigderson, A. One-way functions are essential for non-trivial zero-knowledge. In *[1993] The 2nd Israel Symposium on Theory and Computing Systems* (1993), IEEE, pp. 3–17.

[30] Pietrzak, K. Cryptography from learning parity with noise. In *SOFSEM 2012: Theory and Practice of Computer Science - 38th Conference on Current Trends in Theory and Practice of Computer Science, Špindlerův Mlýn, Czech Republic, January 21-27, 2012. Proceedings* (2012), M. Bieliková, G. Friedrich, G. Gottlob, S. Katzenbeisser, and G. Turán, Eds., vol. 7147 of *Lecture Notes in Computer Science*, Springer, pp. 99–114.

[31] Pietrzak, K., and Sjödin, J. Weak pseudorandom functions in minicrypt. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations* (2008), L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, Eds., vol. 5126 of *Lecture Notes in Computer Science*, Springer, pp. 423–436.

[32] Reingold, O., Trevisan, L., and Vadhan, S. P. Notions of reducibility between cryptographic primitives. In *TCC 2004* (Feb. 2004), M. Naor, Ed., vol. 2951 of *LNCS*, Springer, Heidelberg, pp. 1–20.

[33] Sason, I., and Verdú, S. Bounds among f-divergences. *CoRR abs/1508.00335* (2015).

[34] Simon, D. R. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding* (1998), K. Nyberg, Ed., vol. 1403 of *Lecture Notes in Computer Science*, Springer, pp. 334–345.

[35] Tsybakov, A. B. *Introduction to Nonparametric Estimation*, 1st ed. Springer Publishing Company, Incorporated, 2008.

[36] Vadhan, S. P. Computational entropy. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, O. Goldreich, Ed. ACM, 2019, pp. 693–726.

[37] Vadhan, S. P., and Zheng, C. J. Characterizing pseudoentropy and simplifying pseudorandom generator constructions. In *44th ACM STOC* (May 2012), H. J. Karloff and T. Pitassi, Eds., ACM Press, pp. 817–836.

[38] Valiant, G., and Valiant, P. Estimating the unseen: an n/log(n)-sample estimator for entropy and support size, shown optimal via new clts. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011* (2011), L. Fortnow and S. P. Vadhan, Eds., ACM, pp. 685–694.

[39] Xiao, D. *New Perspectives on the Complexity of Computational Learning, and Other Problems in Theoretical Computer Science*. PhD thesis, Princeton University, 2009.

[40] Xiao, D. Learning to create is as hard as learning to appreciate. In *COLT 2010 - The 23rd Conference on Learning Theory, Haifa, Israel, June 27-29, 2010* (2010), A. T. Kalai and M. Mohri, Eds., Omnipress, pp. 516–528.

[41] Yang, G. *Cryptography and Randomness Extraction in the Multi-Stream Model*. PhD thesis, Tsinghua University, 2016.

Appendix A

Proof of Lemma 1.1.3

. We proceed by presenting a version of the Lemma using mainly Lemma 2.6 in Tsybakov's *Introduction to non-parametric estimation* [35], and a generalization of Tsybakov's result available in e.g. [13].

For the Bhattacharyya coefficient $\sum_{x\in\Omega}\sqrt{p(x)q(x)}$ it holds that

$$\left(\sum_{x\in\Omega}\sqrt{p(x)q(x)}\right)^2 \geq 2^{-KL(p\|q)}.$$

This is a simple consequence of Jensen's inequality. First note that we may write

$$\begin{aligned}\left(\sum_{x\in\Omega}\sqrt{p(x)q(x)}\right)^2 &= 2^{2\log\sum_{x\in\Omega}\sqrt{p(x)q(x)}}\\ &= 2^{2\log\sum_{x\in\Omega}p(x)\sqrt{q(x)/p(x)}}\\ &= 2^{2\log\mathbb{E}_p\left[\sqrt{\frac{q(x)}{p(x)}}\right]}\end{aligned}$$

Noting that log is concave, using Jensen's inequality we obtain that

$$\begin{aligned}2\log\mathbb{E}_p\left[\sqrt{\frac{q(x)}{p(x)}}\right] &\geq 2\,\mathbb{E}_p\left[\log\sqrt{\frac{q(x)}{p(x)}}\right]\\ &= \mathbb{E}_p\left[-\log\frac{p(x)}{q(x)}\right]\\ &= -\mathsf{KL}(p\|q)\end{aligned}$$

To obtain the lemma, note that $\sum_{x\in\Omega}\min(p(x),q(x))+\sum_{x\in\Omega}\max(p(x),q(x)) = 2$ and

$$\mathsf{SD}(p,q) = 1-\sum_{x\in\Omega}\min(p(x),q(x)) \iff \sum_{x\in\Omega}\min(p(x),q(x)) = 1-\mathsf{SD}(p,q),$$

allowing us to bound the Bhattacharyya coefficient using Cauchy-Schwartz: (denote $max := \max(p(x), q(x))$ and $\min := \min(p(x), q(x))$)

$$\begin{aligned}\left(\sum_{x\in\Omega}\sqrt{p(x)q(x)}\right)^2 &= \left(\sum_{x\in\Omega}\sqrt{\max\min}\right)^2 \\ &\leq \sum_{x\in\Omega}\max\sum_{x\in\Omega}\min \\ &= (1+\mathsf{SD}(p,q))(1-\mathsf{SD}(p,q))\end{aligned}$$

Combining this with the bound on relative entropy yields

$$1-\mathsf{SD}(p,q)^2 \geq \left(\sum_{x\in\Omega}\sqrt{p(x)q(x)}\right)^2 \geq 2^{-\mathsf{KL}(p\|q)},$$

which when rearranged gives us

$$\mathsf{SD}(p,q) \leq \sqrt{1-2^{-\mathsf{KL}(p\|q)}} \iff \mathsf{KL}(p\|q) \geq \frac{1}{1-\mathsf{SD}(p,q)^2}.$$

□