BOSTON UNIVERSITY

GRADUATE SCHOOL OF ARTS AND SCIENCES

Dissertation

BUTTERFLY EFFECT IN CRYPTOGRAPHY: CONSEQUENCES OF CHANGES IN DEFINITIONS

by

CHUN-YUAN HSIAO

B.S., National Taiwan University, 1999
M.S., National Taiwan University, 2001

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2010

Approved by

First Reader [illegible]
Leonid Reyzin, Ph.D.
Associate Professor of Computer Science
Boston University

Second Reader [illegible]
Gene Itkis, Ph.D.
Technical Staff of Lincoln Laboratory
Massachusetts Institute of Technology

Third Reader Steven Homer
Steven Homer, Ph.D.
Professor of Computer Science
Boston University

To My Father

Fon-Yu Hsiao

Acknowledgments

I am deeply indebted to my advisor Leonid Reyzin for everything. He has been encouraging, inspiring and patient. I am especially grateful for his help on my presentation skills. Special thanks to Professor Gene Itkis, who created and ran the Applied Cryptography and Electronic Security Group at Boston University Computer Science Department. Thanks also to Professor Steve Homer, Professor Péter Gács, and Professor George Kollios, who served on my Ph.D. examination committee, and to many other Computer Science faculty at Boston University who have helped me over the years in various ways. Thanks to the crypto fellows Nenad, Scott, Robbie, Bhavana, Peng and many other peer graduate students in the department who made my Ph.D. life more enjoyable.

I would like to thank my former advisor Chi-Jen Lu for his continuous support and help even after my Master's degree.

As far as this dissertation is concerned, I would like to thank Yael Tauman Kalai for many helpful discussions, Ron Rivest for assistance with the history of hashing, Vinod Vaikuntanathan for secret-coin hashing examples, and Moni Naor for pointing out related work. Thanks also to the *Crypto 2004* and *Eurocrypt 2007* anonymous referees for insightful comments.

Lastly, I would like to thank my parents and brother. I cannot imagine how I could have finished this long journey without their endless encouragement and support.

BUTTERFLY EFFECT IN CRYPTOGRAPHY: CONSEQUENCES OF CHANGES IN DEFINITIONS

(Order No.)

CHUN-YUAN HSIAO

Boston University Graduate School of Arts and Sciences, 2010

Major Professor: Leonid Reyzin, Associate Professor of Computer Science

ABSTRACT

Modern cryptography places a great deal of emphasis on definitions, because a precise definition formalizes our intuition about a cryptographic primitive.

This dissertation consists of two parts. The first part demonstrates the importance of definitional precision by examining a previously overlooked subtlety in defining a widely-used primitive: the Collision Resistant Hash Function, or CRHF. The subtlety lies in the method by which the CRHF key is generated: namely, whether a trusted party needs to perform key generation (the "secret-coin" variant), or whether any public random string can be used as the key (the "public-coin" variant). Adding a new technique to the so-called "black-box separation" methodology, this thesis shows that these two variants of CRHF, which were sometimes used interchangeably, are actually distinct in general. However, they are also equivalent under certain conditions; the thesis identifies a precise and broad set of such conditions.

The second part of this dissertation investigates two known definitions of entropy. Shannon has shown the equivalence of these two definitions by proving that the shortest compression length of a distribution is equivalent to the amount of randomness it contains. Cryptographers are often interested in distributions that appear random to computationally-bounded observers (for example, ciphertexts often have this property). In an attempt to quantify the amount of this computational randomness, analogues of Shannon's notions of compressibility and entropy have been proposed for the computationally-bounded setting.

Whether these two notions remain equivalent is an interesting open question, with potential applications to pseudorandom generation and cryptographic primitives that rely on it. This thesis shows that they can differ vastly in a common cryptographic setting. One interesting corollary of our work is that we can extract more pseudorandom bits from a distribution if we choose the less commonly used notion of compressibility. In addition to presenting this result, the thesis studies how to better extract pseudorandomness from distributions that are computationally hard to compress.

Contents

List of Abbreviations

CRHF	collision resistant hash function
HILL	Håstad, Impagliazzo, Levin and Luby
MD5	message digest algorithm 5
NIZK	non-interactive zero knowledge
PPTM	probabilistic polynomial-time Turing machine
SHA-1	secure hash algorithm 1

Introduction

Modern cryptography places a great deal of emphasis on definitions, because a precise definition formalizes our intuition about a cryptographic primitive. This dissertation consists of two parts, both demonstrating that a careful choice between two similar definitions is important.

Part One is based on the paper "Finding Collisions on a Public Road, or Do Secure Hash Functions Need Secret Coins?" that appeared in *Crypto 2004*, [28]. Part Two is based on the paper "Conditional Computational Entropy, or Toward Separating Pseudoentropy from Compressibility" that appeared in *Eurocrypt 2007*, [27].

Part I: Secret-Coin vs. Public-Coin

Collision-resistant Hash Function, or CRHF, is one of the earliest primitives of modern cryptography, finding its first uses in digital signatures [45, 46] and Merkle trees [35, 36]. A hash function, of course, maps (potentially long) inputs to short outputs. Informally, a hash function is collision-resistant if it is infeasible to find two inputs that map to the same output. It was first formally defined by [10].

The first part of this dissertation demonstrates the importance of definitional precision by examining a previously overlooked subtlety in defining CRHF. The subtlety lies in the method by which the CRHF key is generated: namely, whether a trusted party needs to perform key generation (the "secret-coin" variant defined in [10]), or whether any public random string can be used as the key (the "public-coin" variant sometimes used in subsequent work). Adding a new technique to the so-called *black-box separation* methodology,[1] we show that these two variants of CRHF, which were sometimes used interchangeably, are actually distinct in general. This oversight has lead to problems, for example, in a proof from [21].

We also show that these two variants are equivalent under certain conditions. In Chapter 4 we study such conditions precisely; here we just remark that there are known examples for both cases: a factoring-based construction shows the equivalence (see Chapter 4), while a lattice-based construction shows the distinction (see [44, 33]).

Part II: Conditional Computational Entropy

The second part of this dissertation investigates two known definitions of entropy. The most common definition, known as Shannon entropy and defined for a distribution X as $\mathbf{E}_{x \in X} - \log_2 \mathbf{Pr}[x]$, is a measure of how much randomness, in terms of number of bits, a distribution contains. Shannon has shown that this measure is equal to the shortest compression length (i.e., the shortest expected number of bits to which elements of X can be compressed) [54]. Thus there are two equivalent ways to define entropy.

Both of the above mentioned entropy definitions are in the information-theoretical setting, meaning no computation constraints are considered. Cryptographers, however, are often interested in distributions that "appear" random to computationally-bounded observers (for example, ciphertexts often have this property). In an attempt to quantify the amount of this "computational" randomness, analogues of Shannon's notions of compressibility and entropy have been proposed for the computationally-bounded setting: indistinguishability based [25] (so-called "HILL entropy") and incompressibility based [3] (so-called "Yao entropy"). Whether these two notions remain equivalent is an interesting open question, with potential applications to pseudorandom generation and cryptographic primitives that rely on it. We show that they can differ vastly in a common cryptographic setting: namely, Yao

[1]Black-box separation methodology was introduced by Impagliazzo and Rudich [32], and it is now a widely-used tool to show how complex a cryptographic primitive is compared to others. See section 5.1 for more details.

entropy can far exceed HILL entropy in a common cryptographic setting.

An important application of the notion of computational entropy is to obtaining pseudorandom bits from distributions. To this end, we need a tool called *extractor*. Extractors were first defined by Nisan and Zuckerman [42] to extract random bits from distributions with entropy, and it has been long known that they can be used to extract pseudorandom bits from distributions with HILL entropy. Barak et al. [3] showed that certain type of extractors can be used to extract pseudorandom bits from distributions with Yao entropy. We re-analyze a well-known extractor construction to show that it satisfies conditions required by the [3] result. In fact, this extractor extracts almost all the computational entropy. Applying this extractor to a distribution with higher Yao entropy (as described in previous paragraph), we show how to extract more pseudorandom bits than possible using the more commonly used HILL entropy.

Part I

Secret-Coin vs. Public-Coin

Chapter 1

Background

Collision-resistant (CR) hashing is one of the earliest primitives of modern cryptography, finding its first uses in digital signatures [45, 46] and Merkle trees [35, 36]. A hash function, of course, maps (potentially long) inputs to short outputs. Informally, a hash function is collision-resistant if it is infeasible to find two inputs that map to the same output.

In practice, such hash functions were constructed to have variable-length inputs mapped to fixed-length outputs. For example, Rabin's hash function [45] has the same output length as a block cipher; MD5 [51] has 128-bit outputs; and SHA-1 [41] has 160-bit outputs. However, it is easy to see there is no meaningful way to formalize the notion of collision-resistance for a single fixed-output-length hash function. Indeed, at least half of the 2^{161} possible 161-bit inputs to SHA-1 [41] have collisions (because SHA-1 has 160-bit outputs). Hence, an algorithm finding collisions for SHA-1 is quite simple: it just has, hardwired in it, two 161-bit strings that collide.

Due to this simple observation, formal definitions of collision-resistant hashing (first given by Damgård [10]) usually speak of collision-resistant function *families* (CRHFs).[1] A hash function family is collision-resistant if any adversary, given a function chosen randomly from the family, is unable to output a collision for it.

[1]It is possible to define a single hash function (with variable output-length; cf. previous paragraph) instead of a collection of them. In this case, it can be collision-resistant only against a uniform adversary.

How to Choose from a Family? Most definitions of CRHFs do not dwell on the issue of *how* a hash function is to be chosen from a family. In the first part of this thesis, we point out that this aspect of the definition is crucial. Indeed, in any application of collision-resistant hashing, some party P must choose a function from the family by flipping some random coins to produce the function description. As we demonstrate, it is important to distinguish between two cases. In the *public-coin* case these random coins can be revealed as part of the function description. In the *secret-coin* case, on the other hand, knowledge of the random coins may allow one to find collisions, and thus P must keep the coins secret after the description is produced. (For examples of both cases, see Chapter 3.) We note that the original definition of [10] is secret-coin, and that the secret-coin definition is more general: clearly, a public-coin CRHF will also work if one chooses to keep the coins secret.

Chapter 2

Motivation and Our Results

2.1 Motivation

Initial Observation The distinction between public-coin and secret-coin CRHFs is commonly overlooked. Some works modify the secret-coin definition of [10] to a public-coin definition, without explicitly mentioning the change (e.g., [4, 55]). Some definitions (e.g., [38]) are ambiguous on this point. This state of affairs leads to confusion and potential problems, as discussed in three examples below.

Example 1. Some applications use the wrong definition of CRHF. For instance, in Zero-Knowledge Sets of Micali, Rabin and Kilian [37], the prover uses a hash function to commit to a set. The hash function is chosen via a shared random string, which is necessary because the prover cannot be trusted to choose his own hash function (since a dishonest prover could benefit from finding collisions), and interaction with the verifier is not allowed at the commit stage (indeed, the prover does not yet know who the verifier(s) will be). In such a setting, one cannot use secret-coin CRHFs (however, in an apparent oversight, [37] defines only secret-coin CRHFs). A clear distinction between public-coin and secret-coin CRHFs would make it easier to precisely state the assumptions needed in such protocols.

Example 2. The result of Simon [55] seems to claim less than the proof implies. Namely, the [55] theorem that one-way permutations are unlikely to imply CRHFs is stated only for public-coin CRHFs, because that is the definition [55] uses. It appears to hold also for secret-coin CRHFs, but this requires re-examining the proof. Such re-examination could be avoided had the definitional confusion been resolved.

Example 3. The original result of Goldwasser and Kalai [21] on the security of the Fiat-Shamir transform without random oracles has a gap due to the different notions of CRHF (the gap was subsequently closed, see below). Essentially, the work first shows that if no *secret-coin* CRHFs exist, then the Fiat-Shamir transform can never work. It then proceeds to show, in a sophisticated argument, that if *public-coin* CRHFs exist, then it is possible to construct a secure identification scheme for which the Fiat-Shamir transform always results in an insecure signature scheme. This gap in the result would be more apparent with proper definitions.

Let us elaborate on the third example, as it was the motivating example for our work. It is not obvious how to modify the [21] proof to cover the case when secret-coin CRHFs exist, but public-coin ones do not. Very recently, Goldwasser and Kalai [20] closed this gap by modifying the identification scheme of the second case to show that the Fiat-Shamir transform is insecure if secret-coin (rather than public-coin) CRHFs exist. Briefly, the modification is to let the honest prover choose the hash function during key generation (instead of the public-coin Fiat-Shamir verifier choosing it during the interaction, as in the earlier version).

Despite the quick resolution of this particular gap, it and other examples above demonstrate the importance of distinguishing between the two types of collision-resistant hashing. Of course, it is conceivable that the two types are equivalent, and the distinction between them is without a difference. We therefore set out to discover whether the distinction between public-coin and secret-coin hashing is real, i.e., whether it is possible that public-coin CRHFs do not exist, but secret-coin CRHFs do.

2.2 Our Results

Recall that public-coin hashing trivially implies secret-coin hashing. We prove the following results:

1. Dense[1] secret-coin CRHFs imply public-coin CRHFs; but
2. There is no black-box reduction from secret-coin CRHFs to public-coin CRHFs.

The first result is quite simple. The second, which is more involved, is obtained by constructing oracles that separate secret-coin CRHFs from public-coin CRHFs. Our technique for this oracle separation is different from previous separations (such as [32, 55, 17, 18, 9]), as explained below. We note that our second result, as most oracle separations, applies only to uniform adversaries (a notable exception to this is [16]).

Our results suggest that a gap between secret-coin and public-coin CRHFs exists, but only if no dense secret-coin CRHFs exist. They highlight the importance of distinguishing between the two definitions of CRHFs.

In addition to these main results, Chapter 6 addresses secret vs. public coins in other cryptographic primitives.

[1]A CRHF is *dense* if a noticeable subset of all keys of a particular length is secure; see Chapter 4.

Chapter 3

Definitions and Notation

Examples Before we define public-coin and secret-coin hashing formally, consider the following two example hash function families. The first one, keyed by a prime p with a large prime $q|(p-1)$, and two elements $g, h \in \mathbb{Z}_p^*$ of order q, computes $H_{p,g,h}(m) = g^{m_1}h^{m_2}$, where m_1 and m_2 are two halves of m (here we think of m as an element of $Z_q \times Z_q$).[1] The second one, keyed by a product n of two primes $p_1 \equiv 3 \pmod 8$, and $p_2 \equiv 7 \pmod 8$ and a value $r \in \mathbb{Z}_n^*$, computes $H_{n,r}(m) = 4^m r^{2^{|m|}} \bmod n$.[2]

The first hash function family is secure as long as discrete logarithm is hard. Thus, if one publishes the random coins used to generate p, g and h, the hash function remain secure (as long as the generation algorithm doesn't do anything esoteric, such as computing h as a random power of g). On the other hand, the second hash function family is secure based on factoring, and is entirely insecure if the factors of n are known. Thus, publishing the random coins used to generate p_1 and p_2 renders the hash function insecure, and the coins must be kept secret.[3]

We say that a function is *negligible* if it vanishes faster than any inverse polynomial. We let PPTM stand for a probabilistic polynomial-time Turing machine. We use $M^?$ to denote

[1]This family is derived from Pedersen commitments [43].

[2]This is essentially the construction of [10] based on the claw-free permutations of [23].

[3]It should be noted, of course, whether it is secure to publish the coins depends not only on the family, but also on the key generating algorithm itself: indeed, the first family can be made insecure if the coins are used to generate h as a power of g, rather than pick h directly. Likewise, the second family could be made secure if it were possible to generate n "directly," without revealing p_1 and p_2 (we are not aware of an algorithm to do so, however).

an oracle Turing machine, and M^A to denote M instantiated with oracle A.

Let k be the security parameter, and let ℓ be a (length) function that does not expand or shrink its input more than a polynomial amount. Below we define two kinds of CRHFs: namely, secret-coin and public-coin. The secret-coin CRHFs definition is originally due to Damgård [10], and the definition here is adapted from [52].

Definition 1. *A* Secret-Coin *Collision Resistant Hash Family is a collection of functions* $\{h_i\}_{i\in I}$ *for some index set* $I \subseteq \{0,1\}^*$*, where* $h_i : \{0,1\}^{|i|+1} \to \{0,1\}^{|i|}$*, and*

1. *There is a PPTM* GEN*, called the generating algorithm, so that* $\mathsf{GEN}(1^k) \in \{0,1\}^{\ell(k)} \cap I$*.*

2. *There exists a PPTM* EVA*, called the function evaluation algorithm, so that* $\forall i \in I$ *and* $\forall x \in \{0,1\}^{|i|+1}$*,* $\mathsf{EVA}(i,x) = h_i(x)$*.*

3. *For all PPTM* ADV*, the probability that* $\mathsf{ADV}(i)$ *outputs a pair* (x,y) *such that* $h_i(x) = h_i(y)$ *is negligible in* k*, where the probability is taken over the random choices of* GEN *in generating* i *and the random choices of* ADV*.*

Definition 2. *A* Public-Coin *Collision Resistant Hash Family is a collection of functions* $\{h_i\}_{i\in\{0,1\}^*}$*, where* $h_i : \{0,1\}^{|i|+1} \to \{0,1\}^{|i|}$*, and*

1. *A PPTM* GEN *on input* 1^k *outputs a uniformly distributed string* i *of length* $\ell(k)$*.*

2. *There exists a PPTM* EVA*, called the function evaluation algorithm, so that* $\forall i \in \{0,1\}^*$ *and* $\forall x \in \{0,1\}^{\ell(|i|)+1}$*,* $\mathsf{EVA}(i,x) = h_i(x)$*.*

3. *For all PPTM* ADV*, the probability that* $\mathsf{ADV}(i)$ *outputs a pair* (x,y) *such that* $h_i(x) = h_i(y)$ *is negligible in* k*, where the probability is taken over the random choices of* GEN *in generating* i *and the random choices of* ADV*.*

A pair (x,y) such that $h_i(x) = h_i(y)$ is called a *collision* for h_i.

Remarks The generating algorithm in the public-coin case is trivially satisfied. We keep it here for comparison with the secret-coin case. Note that in both cases, on security parameter k, GEN outputs a function that maps $\{0,1\}^{\ell(k)+1}$ to $\{0,1\}^{\ell(k)}$. This may seem restrictive as the hash functions only compress one bit. However, it is easy to see that h_i can be extended to $\{0,1\}^n$ for any n, and remain collision-resistant with $\ell(k)$-bit outputs, by the following construction: $h_i^*(x) = h_i(\ldots h_i(h_i(h_i(x_1 \circ x_2 \circ \ldots \circ x_{\ell(k)+1}) \circ x_{\ell(k)+2}) \circ x_{\ell(k)+3}) \ldots \circ x_n)$, where x_j denotes the j-th bit of the input string x.

Chapter 4

Dense Secret-Coin CRHF implies Public-Coin CRHF

The notion of *dense* public-key cryptosystems was introduced by De Santis and Persiano in [11]. By "dense" they mean that a uniformly distributed string, with some noticeable probability, is a secure public key. We adapt the notion of denseness in public-key cryptosystems from [11] to the context of CRHFs. Informally, a d-dense secret-coin CRHF is a secret-coin CRHF with the following additional property: if we pick a k-bit string at random, then we have probability at least k^{-d} of picking an index i for a *collision-resistant* function.[1] Note that, for example, the factoring-based secret-coin CRHF from Chapter 3 is dense, because the proportion of k-bit integers that are products of two equal-length primes is $\Theta(k^{-2})$.

More formally, by h_i being collision-resistant, we mean with probability k^{-d}, i is a "typical" output of GEN. We will use the notion of domination to define typical outputs. Below we cite the definition by Dedić et al. [12].

Definition 3 (g-Domination, [12]). *Let B and C be distributions on the same set S, and g a real-valued function. We say that C g-dominates B if $\forall T \subseteq S$, $Pr_C[T] \geq g(Pr_B[T])$.*

A nice feature about domination is that it preserves over distribution products. Formally,

[1] Confusingly, sometimes the term dense is used to denote a function family where each function has a dense domain, e.g., [24]. This is unrelated to our use of the term.

Lemma 1 ([12]). *If C g-dominates B for a convex function g, then for any distribution D on a set S', $D \times C$ g-dominates $D \times B$.*

With the notion of domination, we are now ready to define dense secret-coin collision-resistant hash fanzine families. Throughout this chapter, let g be a convex polynomial.

Definition 4. *A secret-coin CRHF is d-*dense *if for all k there exists a set S_k in the range of $\mathsf{GEN}(1^k)$ such that $\mathsf{GEN}(1^k)|_{S_k}$ g-dominates $U_{\ell(k)}|_{S_k}$, where $|_S$ means the distribution is conditioned on set S. Furthermore, $\mathbf{Pr}_{i \leftarrow U_{\ell(k)}}[i \in S_k] \geq k^{-d}$, and $\mathbf{Pr}_{i \leftarrow \mathsf{GEN}(1^k)}[i \in S_k] \geq k^{-d'}$ for some integer d'.*

As mentioned in the beginning of the chapter, the factoring-based secret-coin CRHF from Chapter 3 is dense according to our definition. To see this, let S be the set of product of two equal-length primes. The weight of S in the range of GEN is close to one,[2] and the weight of S in $\{0,1\}^{\ell(k)}$ is $\Theta(\ell(k)^{-2})$. $\mathsf{GEN}(1^k)|_{S_k}$ g-dominates $U_{\ell(k)}|_{S_k}$, where g is the identity function, because both of these conditional distributions are uniform.

Constructing public-coin CRHF from dense secret-coin CRHF Given a d-dense secret-coin CRHF, if we pick k^{d+1} strings of length $\ell(k)$ at random, then with high probability, at least one of them defines a collision-resistant hash function.

Hence, we can build a public-coin CRHF from such dense secret-coin CRHF as follows.

1. Generate k^{d+1} random $\ell(k)$-bit strings i_1, i_2, ..., $i_{k^{d+1}}$, independently. These strings specify k^{d+1} hash functions $h_{i_1}, h_{i_2}, \ldots h_{i_{k^{d+1}}}$ in the secret-coin CRHF (strictly speaking, some strings may not define functions at all, because they are not produced by GEN; however, simply define $h_i(x) = 0^{\ell(k)}$ if $\mathsf{EVA}(i, x)$ does not produce an output of length k in the requisite number of steps).

2. Through the construction described in the end of Chapter 3, extend the domain of each of these function to binary strings of length $\ell(k)k^{d+1} + 1$. Let the resulting functions be $\hat{h}_1, \hat{h}_2, \ldots, \hat{h}_{k^{d+1}}$.

[2] GEN has to have real bad coins in order to fail to generate such a product.

3. On an input x of length $\ell(k)k^{d+1}+1$, output the concatenation of $\hat{h}_1(x)$, $\hat{h}_2(x)$, $\ldots$, $\hat{h}_{k^{d+1}}(x)$. Call this function h^*.

The resulting hash h^* maps binary strings of length $\ell(k)k^{d+1}+1$ to binary strings of length $\ell(k)k^{d+1}$, and is collision-resistant because at least one of $\hat{h}_1, \hat{h}_2, \ldots, \hat{h}_{k^{d+1}}$ is a "typical" output of GEN.

The above discussion leads to the following theorem.

Theorem 2. *The existence of dense secret-coin CRHF implies the existence of public-coin CRHF.*

Proof. Assume for contradiction that the resulting public-coin hash family is not collision-resistant. That is, there exists a PPTM adversary A who is able to find a collision for the public-coin hash family. Namely,

$$\mathop{\mathbf{Pr}}_{h^*, A\text{'s coins}}[A \text{ outputs collision for } h^*] = \epsilon$$

and ϵ is not negligible.

By definition of d-dense secret-coin CRHF, there exists a set S in the range of GEN with weight at least $1/k^d$ in $\{0,1\}^{\ell(k)}$, and $\mathsf{GEN}(1^k)|_S$ g-dominates $U_{\ell(k)}|_S$. So the probability that none of the i hits S is $(1-1/k^d)^{k^{d+1}} < e^{-k} < 1/2^k$. Because $\mathbf{Pr}[A$ outputs collision for $h^*]$ is equal to $\mathbf{Pr}[A$ outputs collision for $h^* \mid \exists i \in S] \cdot \mathbf{Pr}[\exists i \in S]$ $+\ \mathbf{Pr}[A$ outputs collision for $h^* \mid \nexists i \in S] \cdot \mathbf{Pr}[\nexists i \in S]$, we have

$$\mathbf{Pr}[A \text{ outputs collision for } h^* \mid \exists i \in S] > \epsilon - 1/2^k.$$

Note that choosing an h^* conditioned on the event that at least one of i is in S is equivalent to choosing k^{d+1} random strings of length $\ell(k)$ such that at least one of them is in S (let's call the uniform distribution on such strings D), and a random $j \in S$, then replacing the first h_i for which $i \in S$ by h_j. Let the resulting hash function be $h^{(1)}$: to select $h^{(1)}$, we choose random element from D and another one from $U_{\ell(k)}|S$. Even though choosing $h^{(1)}$ in this manner instead of simply choosing h^* may seem strange and inefficient, it makes the probability computation easier. We therefore have

$$\mathop{\mathbf{Pr}}_{D\times C\times U_{\ell(k)}|S}[A \text{ outputs collision for } h^{(1)}] \geq \epsilon - 1/2^k ,$$

where C is the distribution of coins needed by A.

Now given a function key α, selected secretly from the dense secret-coin hash family (i.e., $\alpha \leftarrow \mathsf{GEN}(1^k)$), we use A to construct B that finds a collision for α as follows. Generate k^{d+1} uniformly random keys $i_1, \ldots, i_{k^{d+1}}$ of length $\ell(k)$ each. Output "FAIL" if none of these keys is in S; otherwise, replace the first $i \in S$ by α, then give all k^{d+1} keys to A. If A succeeds in funding a collision, then it is a collision for all of the k^{d+1}keys, and, in particular, for α, so B also succeeds in finding a collision. We now need to analyze A's probability of success.

Let the hash function that is given to A be $h^{(2)}$. Note that the success or failure of B depend on a choice of k^{d+1} uniformly random strings, the coins of A (we denote them by C), and α generated by GEN. Since $\mathsf{GEN}(1^k)|S$ g-dominates $U_{\ell(k)}|S$, by Lemma 1 we have

$$\begin{aligned}
&\mathop{\mathbf{Pr}}_{U_{\ell(k)}^{k^{d+1}}\times C\times \mathsf{GEN}(1^k)}[B \text{ outputs collision for } \alpha] \geq \\
&\geq \mathop{\mathbf{Pr}}_{U_{\ell(k)}^{k^{d+1}}\times C\times \mathsf{GEN}(1^k)}[B \text{ outputs collision for } \alpha \mid \exists i \in S] \cdot \mathop{\mathbf{Pr}}_{U_{\ell(k)}^{k^{d+1}}}[\exists i \in S] \\
&\geq \mathop{\mathbf{Pr}}_{D\times C\times \mathsf{GEN}(1^k)}[A \text{ outputs collision for } h^{(2)}] \cdot (1 - 1/2^k) \\
&\geq \mathop{\mathbf{Pr}}_{D\times C\times \mathsf{GEN}(1^k)}[A \text{ outputs collision for } h^{(2)} \mid \alpha \in S] \cdot \mathop{\mathbf{Pr}}_{\mathsf{GEN}(1^k)}[\alpha \in S] \cdot (1 - 1/2^k) \\
&\geq \mathop{\mathbf{Pr}}_{D\times C\times \mathsf{GEN}(1^k)|S}[A \text{ outputs collision for } h^{(2)}] \cdot 1/k^{d'} \cdot (1 - 1/2^k) \\
&\geq g\left(\mathop{\mathbf{Pr}}_{D\times C\times U_{\ell(k)}|S}[A \text{ outputs collision for } h^{(2)}]\right) \cdot 1/k^{d'} \cdot (1 - 1/2^k) \\
&\geq g(\epsilon - 1/2^k) \cdot 1/k^{d'} \cdot (1 - 1/2^k) ,
\end{aligned}$$

which is a contradiction because this probability is not negligible. One last issue is that B may not know where the first $i \in S$ is (to be replaced by α); in fact B may not even know if there exists such an i. But B can simply try α at every location, and this will increase the running time by at most k^{d+1} times. $\square$

Chapter 5

Separating Secret-Coin CRHF from Public-Coin CRHF

5.1 On Oracle Separations

Usually when one constructs a cryptographic primitive P (e.g., a pseudorandom generator [7]) out of another cryptographic primitive Q (e.g., a one-way permutation), P uses Q as a subroutine, oblivious to how Q implemented. The security proof for P usually constructs an adversary for Q using any adversary for P as a subroutine. This is known as a "*black-box reduction* from P to Q."

Note that to show that no *general* reduction from P to Q exists requires proving that Q does not exist, which is impossible given the current state of knowledge. However, it is often possible to show that no *black-box* reduction from P to Q exists; this is important because most cryptographic reductions are black-box.

The first such statement in cryptography is due to Impagliazzo and Rudich [32]. Specifically, they constructed an oracle relative to which key agreement does not exist, but one-way permutations do. This means that any construction of key agreement from one-way permutations does not relativize (i.e., does not hold relative to an oracle). Hence no black-box reduction from key agreement to one-way permutations is possible, because black-box reductions relativize.

The result of [32] was followed by other results about "no black-box reduction from P to Q exists," for a variety of primitives P and Q (e.g., [55, 17, 18, 9]). Most of them, except

[18], actually proved the slightly stronger statement that no *relativizing* reduction from P to Q exists, by using the technique of constructing an oracle.

Our proof differs from most others in that it *directly* proves that no black-box reduction exists, without proving that no relativizing reduction exists. We do so by constructing different oracles for the construction of P from Q and for the security reduction from adversary for P to adversary for Q. This proof technique seems more powerful than the one restricted to a single oracle, although it proves a slightly weaker result. The weaker result is still interesting, however, because it still rules out the most common method of cryptographic reduction. Moreover, the stronger proof technique may yield separations that have not been achievable before.

We note that [18] also directly prove that no black-box reduction exists, without proving that no relativizing reduction exists. Our approach is different from [18], whose approach is to show that for every reduction, there is an oracle relative to which this reduction fails.

For a detailed discussion on black-box reductions, see [48]. All reductions in this paper are what they refer to as *fully black-box* reductions.

5.1.1 Black-Box Reductions

Impagliazzo and Rudich [32] provided an informal definition of black-box reductions, and Gertner et al. [17] formalized it. We recall their formalization.

Definition 5. *A* black-box reduction *from primitive P to primitive Q consists of two oracle PPTMs M and A_Q satisfying the following two conditions:*

If Q can be implemented, so can P: $\forall N$ *(not necessarily PPTM) implementing Q, M^N implements P; and*

If P is broken, so is Q: $\forall A_P$ *(not necessarily PPTM) breaking M^N (as an implementation of P), $A_Q^{A_P,N}$ breaks N (as an implementation of Q).*

The first condition is only a functional requirement; i.e., the term "implement" says nothing about security, but merely says an algorithm satisfies the syntax of the primitive.

5.2 Separating Secret-Coin CRHF from Public-Coin CRHF

5.2.1 The Main Result of Part I

Theorem 3. *There is no black-box reduction from public-coin CRHF to secret-coin CRHF.*

Proof. The following proposition is at the heart of our approach: it shows that it is sufficient to construct different oracles F and G, such that G is used in the *implementations*, while F and G are used for the *adversaries.* This is in contrast to the single-oracle approach usually taken to prove black-box separations.

Proposition 1. *To show that there is no black-box reduction from* public-coin *collision resistant hashing (P) to* secret-coin *collision resistant hashing (Q), it suffices to construct two oracles* F *and* G *such that,*

1. *there is an oracle PPTM L such that $N = L^{\mathsf{G}}$ implements secret-coin hashing;*

2. *for all oracle PPTM M, if M^{G} implements public-coin hashing, then there exists a probabilistic polynomial time adversary A such that $A_P = A^{\mathsf{F}}$ finds a collision for M^{G};*

3. *there is no oracle PPTM B such that $B^{\mathsf{F},\mathsf{G}}$ finds a collision for N.*

Proof. To show that there is no black-box reduction from *public-coin* collision resistant hashing (P) to *secret-coin* collision resistant hashing (Q), we need to negate the definition of black-box reduction from Section 2; i.e., we need to show that for every oracle PPTMs M and A_Q,

Q **can be implemented:** $\exists N$ that implements Q, and if M^N implements P, then

P **can be broken, without breaking** Q**:** $\exists A_P$ that breaks M^N (as an implementation of P), while $A_Q^{A_P,N}$ does not break N (as an implementation of Q).

Recall that "implement" here has only functional meaning.

The first condition clearly implies that Q can be implemented. The second condition also clearly implies that P can be broken: one simply observes that $M^N = M^{L^{\mathsf{G}}}$, and L

is a PPTM; hence, writing M^{G} is equivalent to writing M^N. The third condition implies that P can be broken without breaking Q, essentially because Q can never be broken. More precisely, the third condition is actually stronger than what we need: all we need is that for each A_Q, there is A_P that breaks M^N, while $A_Q^{A_P,N}$ does not break N. Instead, we will show that a single A_P essentially works for all A_Q: namely, $A_P = A^{\mathsf{F}}$, for a fixed oracle F and a polynomial-time A. Such A_P breaks M^N; however, as condition 3 in the proposition statement implies, $A_Q^{A_P,N}$ will be unable to break N, because $A_Q^{A_P,N} = A_Q^{A^{\mathsf{F}},L^{\mathsf{G}}} = B^{\mathsf{F},\mathsf{G}}$ for some oracle PPTM B. $\square$

Remarks Note that if the implementation has access to not only G but also F, it becomes the usual single-oracle separation. The reason why we do not give the implementation access to F is to avoid "self-referencing" when defining F. To see this, note that F is the "collision finder" and is defined according to the oracles that the implementation has access to.[1]

The rest of this section is devoted to constructing such F and G and proving that they work.

5.2.2 The Oracles F and G

In constructing F and G, we will use the Borel-Cantelli Lemma (see, e.g., [2]), which states that if the sum of the probabilities of a sequence of events converges, then the probability that infinitely many of these events happen is zero. Formally,

Lemma 4 (Borel-Cantelli Lemma). *Let $B_1, B_2, \ldots$ be a sequence of events on the same probability space. Then $\sum_{n=1}^{\infty} \mathbf{Pr}[B_n] < \infty$ implies $\mathbf{Pr}[\bigwedge_{k=1}^{\infty} \bigvee_{n \geq k} B_n] = 0$.*

We first construct "random" F (collision-finder) and G (secret-coin hash), and then use the above lemma to show that at least one pair of F and G works.

Intuitively, we want F to break any public-coin hashing but not break some secret-coin hashing. More precisely, F will find a collision if it is supplied with the coins of the generating algorithm and will refuse to do so without the coins.

[1]Similar concern occurs in [55], where constructing the collision-finder requires more careful design.

- G consists of two collections of functions $\{g_i\}_{i\in\mathbb{N}}$ and $\{h_\alpha\}_{\alpha\in\{0,1\}^*}$, where each g_i is a random function from $\{0,1\}^i$ to $\{0,1\}^{2i}$. We will call a binary string *valid* if it is in the range of g, and *invalid* if not. Each h_α is a random function from $\{0,1\}^{|\alpha|+1}$ to $\{0,1\}^{|\alpha|}$ if α is valid, and is a constant function $0^{|\alpha|}$ if α is invalid. We will call queries to h_α valid (resp. invalid) if α is valid (resp. invalid).

- F takes a deterministic oracle machine $M^?$ and 1^ℓ as input, and outputs a collision of length $\ell+1$ for M^{G} if M^{G} satisfies the following conditions.

  1. M^{G} maps $\{0,1\}^{\ell+1}$ to $\{0,1\}^\ell$.
  2. M^{G} never queries h_α for some α not obtained by previously querying g. I.e., whenever M^{G} queries h_α, this α is the answer to some g-query that M^{G} has previously asked.

  When both conditions hold, F picks a random x from $\{0,1\}^{\ell+1}$ that has a collision, then a random y ($\neq x$) that collides to x (i.e., $M^{\mathsf{G}}(x) = M^{\mathsf{G}}(y)$), and outputs (x,y). Otherwise F outputs $\perp$.

  Observe that when F outputs (x,y), not only x, but also y is uniformly distributed over all points that have a collision. Indeed, let C be the total number of points that have a collision, and suppose y has c collisions $(x_1,x_2,\ldots,x_c)$: then $\mathbf{Pr}[y \text{ is chosen}] = \sum_{i=1}^{c} 1/c\,\mathbf{Pr}[x_i \text{ is chosen}] = 1/c\cdot(c/C) = 1/C$.

Remarks The reason for g being length-doubling is to have a "sparse" function family. More specifically, it should be hard to get a value in the range of g without applying it.

As in [55], there are various ways of constructing F (the collision-finding oracle): one can choose a random pair that collides, or a random x then a random y (possibly equal to x) that collides to x. The second construction has the advantage, in analysis, that both x and y are uniformly distributed but does not always give a "correct" collision, like the first one does. Our F has both properties.

5.2.3 Secret-Coin Collision-Resistant Hash Family Based on G

In this section we construct a secret-coin CRHF. The construction is straightforward given the oracle G: the generating algorithm uses g and the hashing uses h. More precisely, on input 1^k the generating algorithm picks a random seed $r \in \{0,1\}^k$ and outputs $\alpha = g_k(r)$. The hash function is h_α. Note that the adversary A (who is trying to find a collision) is given only α but not r. We will show that for measure one of oracles F and G, the probability over r and A's coin tosses that A finds a collision for h_α is negligible. Recall that A has access to both F and G.

Define D as the event that A outputs a collision for h_α in the following experiment:

$$r \leftarrow_R \{0,1\}^k,\ \alpha \leftarrow g_k(r),\ (x,y) \leftarrow A^{\mathsf{F},\mathsf{G}}(\alpha).$$

And in the same experiment, define B as the event that during its computation, A queries F on $M^?$, where $M^?$ is some deterministic oracle machine that queries its oracle on a preimage of α under g_k (i.e., intuitively, $M^?$ has r hardwired in it). Suppose A's running time is bounded by k^c for some constant c. The probability that B happens is at most the probability of inverting the random function g_k. If α has a unique preimage, this is at most $k^c/2^k$; the probability that α has two or more preimages is at most $1/2^k$ (because it's the probability that r collides with another value under g_k); hence $\mathbf{Pr}[B] \le (k^c+1)/2^k$. The probability that D happens conditioned on $\neg B$ is at most the probability of finding a collision for random function h_α, which is bounded by $k^{2c}/2^{2k}$. Recall that A can be randomized. We thus have

$$\begin{aligned}
\mathop{\mathbf{Pr}}_{\mathsf{F},\mathsf{G},r,A}[D] &= \mathbf{Pr}[B]\cdot\mathbf{Pr}[D|B] + \mathbf{Pr}[\neg B]\cdot\mathbf{Pr}[D|\neg B] \\
&\le \mathbf{Pr}[B] + \mathbf{Pr}[D|\neg B] \\
&\le (k^c+1)/2^k + k^{2c}/2^{2k} \\
&\le 2k^c/2^k\,.
\end{aligned}$$

By the Markov inequality, $\mathbf{Pr}_{\mathsf{F},\mathsf{G}}[\mathbf{Pr}_{r,A}[D] \ge k^2\cdot 2k^c/2^k] \le 1/k^2$. Since $\sum_k 1/k^2$ converges, the Borel-Cantelli lemma implies that for only measure zero of F and G, can there be infinitely many k for which event D happens with probability (over r and A's coins) greater than or equal to $k^{c+2}/2^{k-1}$. This implies that for measure one of F and G, event D happens with probability (over r and A's coins) smaller than $k^{c+2}/2^{k-1}$ (a negligible function) for all large enough k. There are only countably many adversaries A, so we have the following lemma.

Lemma 5. *For measure one of* F *and* G*, there is a CRHF using* G*, which is secure against adversaries using* G *and* F*.*

5.2.4 No Public-Coin Collision-Resistant Hash Family Based on G

In this section we show that any implementation of public-coin hashing using oracle G cannot be collision-resistant against adversaries with oracle access to both F and G.[2] More precisely, let $r \in \{0,1\}^{\ell(k)}$ be the public randomness used by the generating algorithm for a family of hash functions, and let $M^{?}$ be the evaluation algorithm. I.e., $M^{\mathsf{G}}(r,\cdot)$ is the hash function specified by r. Assume that $M_r^{\mathsf{G}}(\cdot) \triangleq M^{\mathsf{G}}(r,\cdot)$ maps $\{0,1\}^{\ell(k)+1}$ to $\{0,1\}^{\ell(k)}$, where ℓ is a function that does not expand or shrink the input by more than a polynomial amount. We will show how to find x and y of length $\ell(k)+1$ such that $M_r^{\mathsf{G}}(x) = M_r^{\mathsf{G}}(y)$.

An immediate attempt is to query $\mathsf{F}(M_r^{?}, 1^{\ell(k)})$, but notice that M_r^{G} may query h_α for arbitrary α,[3] which prevents F from finding a collision for us. However, these α are likely to be invalid, and hence oracle answers to these queries are likely to be $0^{|\alpha|}$. So we can construct a machine $\tilde{M}_r^{?}$ that behaves "similar" to $M_r^{?}$ but only after getting α from g does it query h_α. And instead of finding collision for M_r^{G}, we find collision for $\tilde{M}_r^{\mathsf{G}}$, which can be done by simply querying $F(\tilde{M}_r^{?}, 1^{\ell(k)})$.

Suppose the running time of M_r^{G} is bounded by k^c for some constant $c > 1$. Before simulating M_r^{G}, $\tilde{M}_r^{\mathsf{G}}$ queries g on all inputs of length smaller than or equal to $4c \log k$. This takes $2k^{4c}$ steps. Now $\tilde{M}_r^{\mathsf{G}}$ simulates M_r^{G} step by step, except for queries to h_α. If α is the answer to one of the queries $\tilde{M}_r^{\mathsf{G}}$ already asked of G (either before the beginning of the

[2]In fact, only F is needed to find a collision.

[3]In particular, those α not obtained by previously querying g.

simulation or when simulating M_r^{G}), then $\tilde{M}_r^{\mathsf{G}}$ actually queries h_α. Else it returns $0^{|\alpha|}$ as the answer to M_r^{G} without querying h_α.

Now fix r and x. For every $M^?$ the probability, over random G, that $\tilde{M}_r^{\mathsf{G}}(x) \neq M_r^{\mathsf{G}}(x)$ is at most the probability, over G, that M_r^{G} queries h_α for some valid α of length greater than $8c \log k$ without receiving it from g.[4] Consider the very first time that M_r^{G} makes such a "long" valid query. Let n_g be the number of queries to g on inputs longer than $4c \log k$, and n_h be the number of invalid queries to h prior to this point. Then the probability in question is upper bounded by $k^c \cdot \frac{k^{4c}-n_g-n_h}{k^{8c}-n_g}$, which is at most $1/k^{3c}$. For every fixed G and r, call an x "bad" if $\tilde{M}_r^{\mathsf{G}}(x) \neq M_r^{\mathsf{G}}(x)$. We have

$$\mathop{\mathbf{E}}_{\mathsf{G}}[\mathop{\mathbf{Pr}}_{x}[x \text{ is bad}]] = \mathop{\mathbf{Pr}}_{\mathsf{G},x}[x \text{ is bad}] \leq 1/k^{3c}.$$

Next, notice that there are at most half of x that have no collisions, and F would pick its answer $(x_{\mathsf{F}}, y_{\mathsf{F}})$, uniformly, from those points that have a collision. So for a fixed G, the probability over F that x_{F} is bad is at most twice the probability over random $x \in \{0,1\}^{\ell(k)+1}$ that x is bad. Also recall that the distribution of y_{F} is the same as x_{F}. So for every $M^?$,

$$\mathop{\mathbf{E}}_{\mathsf{G}}[\mathop{\mathbf{Pr}}_{\mathsf{F}}[\text{at least one of } (x_{\mathsf{F}}, y_{\mathsf{F}}) \text{ is bad}]] \leq 4 \cdot \mathop{\mathbf{E}}_{\mathsf{G}}[\mathop{\mathbf{Pr}}_{x}[x \text{ is bad}]].$$

If none of $(x_{\mathsf{F}}, y_{\mathsf{F}})$ is bad, this pair would be a collision not only for $\tilde{M}_r^{\mathsf{G}}$ but also for M_r^{G}. We have

$$\mathop{\mathbf{Pr}}_{F,\mathsf{G},r}[(x_{\mathsf{F}}, y_{\mathsf{F}}) \text{ is not a collision of } M_r^{\mathsf{G}}] \leq 4 \mathop{\mathbf{Pr}}_{\mathsf{G},x,r}[x \text{ is bad}] \leq 4/k^{3c},$$

then

$$\mathop{\mathbf{Pr}}_{\mathsf{F},\mathsf{G}}[\mathop{\mathbf{Pr}}_{r}[(x_{\mathsf{F}}, y_{\mathsf{F}}) \text{ is not a collision of } M_r^{\mathsf{G}}] \geq 4/k^c] \leq 1/k^{2c}.$$

Since $\sum_k 1/k^{2c}$ converges, the Borel-Cantelli lemma implies that for only measure zero of F and G, can we have $\mathbf{Pr}_r[(x_{\mathsf{F}}, y_{\mathsf{F}})$ is not a collision of $M_r^{\mathsf{G}}] \geq 4/k^c$ for infinitely many k. In other words, for measure one of F and G, $\mathbf{Pr}_r[(x_{\mathsf{F}}, y_{\mathsf{F}})$ is a collision of $M_r^{\mathsf{G}}] \geq 4/k^c$ for all

[4]Recall that g is length-doubling.

large enough k. There are only countably many oracle machines $M^?$, each of which can be collision resistant for only measure zero of F and G. We conclude the following.

Lemma 6. *For measure one of* F *and* G*, any implementation of public-coin hash function families using* G *cannot be collision-resistant against adversaries using* F.

This concludes the proof of Theorem 3. $\square$

Chapter 6

Other Primitives

Public Coins vs. Secret Coins For Other Primitives Perhaps the lack of attention in the literature to the distinction between secret- and public-coin primitives is due, in part, to the fact that this distinction is often not meaningful.

For example, for one-way function families, these two notions are equivalent, because a secret-coin one-way function family implies a single one-way function (which trivially implies a public-coin one-way function family). Indeed, take the generating algorithm g and evaluation algorithm f and define $F(r,x) \triangleq (g(r), f_{g(r)}(x))$; this is one-way because an adversary who can come up with (r', x') such that $g(r) = g(r')$ and $f_{g(r')}(x') = f_{g(r)}(x)$ can be directly used to invert $f_{g(r)}(x)$, since $f_{g(r)}(x') = f_{g(r')}(x') = f_{g(r)}(x)$.

On the other hand, for trapdoor permutations (and public-key schemes), the notion of public-coin generation is meaningless: indeed the trapdoor (or the secret key) must be kept secret.

However, it seems that this distinction is interesting for some primitives in addition to collision-resistant hash functions. The relationships between public-coin and secret-coin versions of one-way permutation families and claw-free permutation families are unknown.[1] In particular, claw-free permutations are related to collision-resistant hashing [10, 52], which suggests that the distinction for claw-free permutations is related to the distinction for CRHFs.

[1]We believe that the same construction of F and G (up to slight modifications) separates public-coin and secret-coin one-way permutation families.

Part II

Conditional Computational Entropy

Chapter 7

Background

The various information-theoretic definitions of entropy measure the amount of randomness a probability distribution has. As cryptography is able to produce distributions that appear, for computationally bounded observers, to have more randomness than they really do, various notions of *computational* entropy attempt to quantify this *appearance* of entropy. The commonly used HILL entropy (so named after [25]) says that a distribution has computational entropy k if it is indistinguishable (in polynomial time) from a distribution that has information-theoretic entropy k.[1] The so-called Yao entropy [61, 3], says that a distribution has computational entropy k if it cannot be efficiently compressed to below k bits and then efficiently decompressed. Other computational notions of entropy have been considered as well [3, 25].

Computational notions of entropy are useful, in particular, for extracting strings that are pseudorandom (i.e., look uniform to computationally bounded observers) from distributions that appear to have entropy. Indeed, generation of pseudorandom bits is the very purpose of computational entropy defined in [25], and its variant considered in [15]. Pseudorandom bits have many uses, for example, as keys in cryptographic applications.

[1] The specific notion of information-theoretic entropy depends on the desired application; for the purposes of this paper, we will use min-entropy, defined in Chapter 9.

Chapter 8

Our Results

The adversary in cryptographic applications (or, more generally, an observer) often possesses information related to the distribution whose entropy is being measured. For example, in the case of Diffie-Hellman key agreement [13] the adversary has g^x and g^y, and the interesting question is the amount of computational entropy of g^{xy}. Thus, the entropy of a distribution for a particular observer (and thus the pseudorandomness of the extracted strings) depends on what other information the observer possesses. Because notions of computational entropy necessarily refer to computationally-bounded machines (e.g., the distinguisher for the HILL entropy or the compressor and decompressor for the Yao entropy), they must also consider the information available to these machines. This has sometimes been done implicitly (e.g., in [15]); however, most commonly used definitions do not do so explicitly.

In this work, we explicitly put forward notions of *conditional* computational entropy. This allows us to:

1. Separate conditional Yao entropy from conditional HILL entropy by demonstrating a joint distribution (X, Z) such that X has high Yao entropy but low HILL entropy when conditioned on Z.

2. Demonstrate (to the best of our knowledge, first) application of Yao entropy by extracting more pseudorandom bits from a distribution using Yao-entropy-based techniques than seems possible from HILL-entropy-based techniques.

3. Define a new, natural notion of unpredictability entropy, which can be used, in particular, to talk about the entropy of a value that is unique, such as g^{xy} where g^x and g^y are known to the observer, and possibly even verifiable, such as the preimage x of a one-way permutation f, where $y = f(x)$ is known to the observer.

HILL-Yao Separation. The first contribution (Section 10.1) can be seen as making progress toward the open question of whether Yao entropy implies HILL entropy, attributed in [59] to Impagliazzo [31] (the converse is known to be true: HILL entropy implies Yao entropy, because compressibility implies distinguishability). Wee [60] showed that Yao entropy does not imply HILL entropy in the presence of a random oracle and a membership testing oracle. Our separation of conditional Yao entropy from conditional HILL entropy can be seen as an improvement of the result of [60]: it shows that Yao entropy does not imply HILL entropy in the presence of a (short) random string, because the distribution Z on which X is conditioned is simply the uniform distribution on strings of polynomial length. The separation holds under the quadratic residuosity assumption.

Randomness Extraction. Usually, pseudorandomness extraction is analyzed via HILL entropy, because distributions with HILL entropy are indistinguishable from distributions with the same statistical entropy, and we have tools (namely, randomness extractors [42]) to obtain uniform strings from the latter. Tools are also available to extract from Yao entropy: namely, extractors with a special *reconstruction* property [3]. Our second contribution (Section 11.3) is to show that considering the Yao entropy and applying a reconstructive extractor can yield many more pseudorandom bits than the traditional analysis, because, according to our first result, Yao entropy can be much higher than HILL entropy. This appears to be the first application of Yao entropy, and also demonstrates the special power of reconstructive extractors.

It is worth mentioning that while our separation of entropies is conditional, the extraction result holds even for the traditional (unconditional) notion of pseudorandomness.

The analysis of pseudorandomness of the resulting string, however, relies on the notion of conditional entropy, thus demonstrating that it can be a useful tool even in the analysis of pseudorandomness of unconditional distributions.

Unpredictability Entropy. Unpredictability entropy is a natural formalization of a previously nameless notion that was implicitly used in multiple works. Our definition essentially says that if some value cannot be predicted from other information with probability higher than 2^{-k}, then it has entropy k when conditioned on that information. For example, when a one-way permutation f is hard to invert with probability higher than 2^{-k}, then conditioned on $f(x)$, the value x has entropy k. The use of *conditional* entropy is what makes this definition meaningful for cryptographic applications.

We demonstrate that almost k pseudorandom bits can be extracted from distributions with unpredictability entropy k, by showing that unpredictability entropy implies conditional Yao entropy, to which reconstruction extractors can be applied. Thus, unpredictability entropy provides a simple language that allows, in particular, known results on hardcore bits of one-way functions to be stated more generally.

We also prove other (fairly straightforward) relations between unpredictability entropy and HILL and Yao conditional entropies.

Chapter 9

Definitions and Notation

In this section we recall the HILL and Yao definitions of computational entropy (or pseudoentropy) and provide the new, conditional definitions.

Notation. We will use n for the length parameter; our distributions will be on strings of length polynomial in n. We will use s as the circuit size parameter (or running time bound when dealing with Turing machines instead of circuits). To denote a value x sampled from a distribution X, we write $x \leftarrow X$. We denote by $M(X)$ the probability distribution on the outputs of a Turing machine M, taken over the coin tosses (if any) of M and the random choice of the input x according to the distribution X. We use U_n to denote the uniform distribution on $\{0,1\}^n$. For a joint distribution (X, Z), we write X_z to denote the conditional distribution of X when $Z = z$; conversely, given a collection of distributions X_z and a distribution Z, we use (X, Z) to denote the joint distribution given by $\mathbf{Pr}[(X, Z) = (x, z)] = \mathbf{Pr}[Z = z]\,\mathbf{Pr}[X_z = x]$.

We may describe more complicated distributions by describing the sampling process and then the sampled outcome. For example, $\{a \leftarrow X; b \leftarrow X : (a, b)\}$ denotes two independent samples from X, while $\{a \leftarrow X : (a, M(a, Y))\}$ denotes the distribution obtained by sampling X to get a, sampling Y to get b, running $M(a, b)$ to get c, and outputting (a, c).

The statistical distance between two distributions X and Y, denoted by $\texttt{dist}(X, Y)$, is defined as $\max_T |\mathbf{Pr}[T(X) = 1] - \mathbf{Pr}[T(Y) = 1]|$ where T is any test (function). (This is equivalent to the commonly seen $\mathtt{dist}(X, Y) = \frac{1}{2}\sum_a |\mathbf{Pr}[X = a] - \mathbf{Pr}[Y = a]|$.) The computational distance with respect to size s circuits, denoted by $\mathtt{cdist}_s(X, Y)$, limits T to be any circuit of size s.

Unconditional Computational Entropy. The min-entropy of a distribution X, denoted by $\mathbf{H}_\infty(X)$, is defined as $-\log(\max_x \mathbf{Pr}[X = x])$. Although min-entropy provides a rather pessimistic view of a distribution (looking only at its worst-case element), this notion is useful in cryptography, because even a computationally unbounded predictor can guess the value of a sample from X with probability at most $2^{-\mathbf{H}_\infty(X)}$. Most results on randomness extractors are formulated in terms of min-entropy of the source distribution.

The first definition says that a distribution has high computational min-entropy if it is *indistinguishable* from some distribution with high statistical min-entropy. It can thus be seen as generalization of the notion of pseudorandomness of [61], which is defined as indistinguishability from uniform.

Definition 6 ([25, 3]). *A distribution X has* **HILL entropy** *at least k, denoted by $\mathbf{H}^{\mathsf{HILL}}_{\epsilon,s}(X) \geq k$, if there exists a distribution Y such that $\mathbf{H}_\infty(Y) \geq k$ and $\mathtt{cdist}_s(X, Y) \leq \epsilon$.*

(In [25] Y needs to be efficiently samplable; however, for our application, as well as for [3], samplability is not required.)

Another definition of computational entropy considers compression length. Shannon's theorem [54] says that the minimum compression length of a distribution, by all possible compression and decompression functions, is equal to its average entropy (up to small additive terms). Yao [61] proposed to measure computational entropy by imposing computational constraints on the compression and decompression algorithms.[1] In order to convert this into a worst-case (rather than average-case) metric similar to min-entropy, Barak et al. [3] require that any subset in the support of X (instead of only the entire X) be hard to compress.

[1]Yao called it "effective" entropy.

Definition 7 ([61, 3]). *A distribution X has* **Yao entropy** *at least k, denoted by $\mathbf{H}^{\mathsf{Yao}}_{\epsilon,s}(X) \geq k$, if for every pair of circuits c, d (called "compressor" and "decompressor") of total size s with the outputs of c having length ℓ,*

$$\Pr_{x \leftarrow X}[d(c(x)) = x] \leq 2^{\ell - k} + \epsilon.$$

Note that just like HILL entropy, for $\epsilon = 0$ this becomes equivalent to min-entropy (this can be seen by considering the singleton set of the most likely element).

Conditional Computational Entropy. Before we provide the new conditional definitions of computational entropy, we need to consider the information-theoretic notion of conditional min-entropy.

Let (Y, Z) be a distribution. If we take the straightforward average of the min-entropies $\mathbf{E}_{z \leftarrow Z}[\mathbf{H}_\infty(Y_z)]$ to be the conditional min-entropy, we will lose the relation between min-entropy and prediction probability, which is important for many applications (see e.g. Lemma 11 and Lemma 16). For instance, if for half of Z, $\mathbf{H}_\infty(Y_z) = 0$ and the other half $\mathbf{H}_\infty(Y_z) = 100$, then, given a random z, Y can be predicted with probability over $1/2$, much more than 2^{-50} the average would suggest. A conservative approach, taken in [50], would be to take the minimum (over z) of $\mathbf{H}_\infty(Y_z)$. [2] However, this definition may kill "good" distributions like $Y_z = U_n$ for all $z \neq 0^n$ and $Y_z = 0^n$ for $z = 0^n$; although this problem can be overcome by defining a so-called "smooth" version [50, 49], we follow a different approach.

For the purposes of randomness extraction, Dodis et al. [14] observed that because Z is not under adversarial control, it suffices that the *average*, over Z, of the maximum probability is low. They define average min-entropy: $\tilde{\mathbf{H}}_\infty(Y|Z) \stackrel{\text{def}}{=} -\log(\mathbf{E}_{z \leftarrow Z}[2^{-\mathbf{H}_\infty(Y|Z=z)}] = -\log(\mathbf{E}_{z \leftarrow Z}[\max_y \Pr[Y_z = y]])$. This definition averages prediction probabilities before taking the logarithm and ensures that for any predictor P, $\Pr_{(y,z) \leftarrow (Y,Z)}[P(z) = y] \leq 2^{-\tilde{\mathbf{H}}_\infty(Y|Z)}$. It also ensures that randomness extraction works almost as well as it does for unconditional distributions; see Section 11.1.

[2]For some applications, e.g. [15, 26], this rather stringent condition can be met.

Using this definition of conditional min-entropy, defining conditional HILL-entropy is straightforward.

Definition 8 (Conditional HILL entropy). *For a distribution (X, Z), we say X has HILL entropy at least k conditioned on Z, denoted by $\mathbf{H}^{\mathsf{HILL}}_{\epsilon,s}(X|Z) \geq k$, if there exists a collection of distributions Y_z (giving rise to a joint distribution (Y, Z)) such that $\tilde{\mathbf{H}}_\infty(Y|Z) \geq k$ and $\mathtt{cdist}_s((X, Z), (Y, Z)) \leq \epsilon$.*

For conditional Yao entropy, we simply let the compressor and decompressor have z as input.

Definition 9 (Conditional Yao entropy). *For a distribution (X, Z), we say X has Yao entropy at least k conditioned on Z, denoted by $\mathbf{H}^{\mathsf{Yao}}_{\epsilon,s}(X|Z) \geq k$, if for every pair of circuits c, d of total size s with the outputs of c having length ℓ,*

$$\mathop{\mathbf{Pr}}_{(x,z)\leftarrow(X,Z)}[d(c(x, z), z) = x] \leq 2^{\ell-k} + \epsilon.$$

We postpone the discussion of unpredictability entropy until Section 12.

Asymptotic Definitions. All above definitions are with respect to a single distribution and fixed-size circuits. We are also interested in their asymptotic behaviors, so we consider *distribution ensembles*. In this case, everything is parameterized by n: $X^{(n)}$, $s(n)$, and $\epsilon(n)$. In such a case, whether circuits in our definitions are determined after n is chosen (the nonuniform setting), or whether an algorithm of running time $s(n)$ is chosen independent of n (the uniform setting) makes a difference. We consider the nonuniform setting.

We omit the subscripts $s(n)$ and $\epsilon(n)$ when they "denote" any polynomial and negligible functions, respectively ($\epsilon(n)$ is negligible if $\epsilon(n) \in n^{-\omega(1)}$). More precisely, we write $\mathbf{H}^{\mathsf{HILL}}(X^{(n)}) \geq k(n)$, if there is a distribution ensemble $Y^{(n)}$ such that $\mathbf{H}_\infty(Y^{(n)}) \geq k(n)$ for all n, and for every polynomial $s(n)$, there exists a negligible $\epsilon_s(n)$ such that $\mathtt{cdist}_{s(n)}(X^{(n)}, Y^{(n)}) \leq \epsilon_s(n)$. Similarly for the other definitions.

Chapter 10

Separating HILL Entropy from Yao Entropy

10.1 Separating HILL Entropy from Yao Entropy

In this section we construct a joint distribution (X, Z),[1] such that given Z, the distribution X has high Yao but low HILL entropy; namely, $\mathbf{H}^{\mathsf{Yao}}(X|Z) \gg \mathbf{H}^{\mathsf{HILL}}(X|Z)$. This is a separation of conditional HILL and Yao entropies. Since Z will be simply a polynomially long random string, this result can also be viewed as a separation of Yao entropy and HILL entropy in the Common Reference String (CRS) model. (In this model one assumes that a uniformly-distributed string of length $q(n)$, for some fixed polynomial q, is accessible to everyone.)

Our construction uses a non-interactive zero knowledge proof system, so we describe it briefly in the following section.

10.2 Non-Interactive Zero Knowledge (NIZK)

NIZK was introduced by Blum et al. [6, 5]. For our purposes, a single-theorem variant suffices. Let λ be a positive polynomial and $L \in \mathcal{NP}$ be a language that has witnesses of length n for theorems of lengths $(\lambda(n-1), \lambda(n)]$. (It is easier for us to measure everything in terms of witness length rather than the more traditional theorem length, but they are

[1] Actually, (X, Z) should be defined as a distribution ensemble $(X^{(n)}, Z^{(n)})$, but we'll omit the superscript for ease of notation.

anyway polynomially related for the languages we are interested in.) NIZK works in the CRS model. Let q be a positive polynomial, and let the CRS be $r \leftarrow U_{q(n)}$ when witnesses are of length n. A NIZK proof system for L is a pair of polynomial-time Turing machines (P, V), called the *prover* and the *verifier* (as well as the polynomial q) such that the following three conditions hold.

1. Completeness: $\forall \phi \in L$ with NP witness w, if $\pi = \mathsf{P}(\phi, w, r)$ is the proof generated by P, then $\mathbf{Pr}_{r \leftarrow U_{q(n)}}[\mathsf{V}(\phi, \pi, r) = 1] = 1$.[2]

2. Soundness: Call r *bad* if $\exists \phi \notin L$, $\exists \pi'$, such that $\mathsf{V}(\phi, \pi', r) = 1$ (and *good* otherwise). Then $\mathbf{Pr}_{r \leftarrow U_{q(n)}}[r \text{ is bad}]$ is negligible in n.

3. Zero-knowledgeness: There is a probabilistic polynomial time Turing machine SIM called the simulator, such that for every $\phi \in L$ and every witness w for ϕ, $\mathsf{SIM}(\phi) = (\phi, \Pi_{\mathsf{SIM}}, R_{\mathsf{SIM}})$ is computationally indistinguishable from $(\phi, \Pi, R) = \{r \leftarrow U_{q(n)}\ ;\ \pi \leftarrow \mathsf{P}(\phi, w, r) : (\phi, \pi, r)\}$.

For our analysis, we need two additional properties. First, we need the proofs π not to add too much entropy. For this, we use ideas on unique NIZK by Lepinski, Micali and shelat [34]. We do not need the full-fledged uniZK system; rather, the single-theorem system described as the first part of the proof of [34, Theorem 1] suffices (it is based on taking away most of the prover freedom for the single-theorem system of [5]). The protocol of [34] is presented in the public-key model, in which the prover generates the public key (x, y) consisting of an n-bit modulus x and n-bit value $y \in \mathbb{Z}_x^*$. To make it work for our setting, we simply have the prover generate the public key during the proof and put it into π. Once the public key is fixed, the prover has no further choices in generating π, except choosing a witness w for $\phi \in L$ (note that this actually requires a slight modification to the proof of [34], which we describe in Appendix A).

The second property we need is that the simulated shared randomness R_{SIM} is independent of the simulator input ϕ. It is satisfied by the [34] proof system (as well as by the [5]

[2]If P is probabilistic, the probability is taken over the choice r and random choices made by P.

system on which it is based).

The zero-knowledge property of the [34] proof system is based on the following assumption (the other properties are unconditional).

Assumption 1 (Quadratic Residuousity [22] for Blum Integers). *For all probabilistic polynomial time algorithms P, if p_1 and p_2 are random $n/2$-bit primes congruent to 3 modulo 4, y is a random integer between 1 and p_1p_2 with Jacobi symbol $\left(\frac{y}{p_1p_2}\right) = 1$, and $b = 1$ if y is a quadratic residue modulo p_1p_2 and 0 otherwise, then $|1/2 - \mathbf{Pr}[P(y, p_1p_2) = b]|$ is negligible in n.*

The formal statement of the properties we need from [34] follows.

Lemma 7 ([34]+Appendix A). *If the above assumption holds, then there exists an NIZK proof system for any language $L \in \mathcal{NP}$ with the following additional properties: (1) if r is good and ϕ has t distinct witnesses w, then the number of proofs π for ϕ that are accepted by V is at most $t2^{2n}$, and (2) the string R_{SIM} output by the simulator is independent of the simulator input ϕ.*

10.3 The Construction

Our intuition is based on the separation by Wee [60], who demonstrated an oracle relative to which there is a random variable that has high Yao and low HILL entropy. His oracle consists of a random length-increasing function and an oracle for testing membership in the sparse range of this function. The random variable is simply the range of the function. The ability to test membership in the range helps distinguish it from uniform, hence HILL entropy is low. On the other hand, knowing that a random variable is in the range of a random function does not help to compress it, hence Yao entropy is high.

We follow this intuition, but replace the length-increasing random function and the membership oracle with a pseudorandom generator and an NIZK proof of membership, respectively. Our distribution X consists of two parts: 1) output of a pseudorandom generator and, 2) an NIZK proof that the first part is as alleged. However, an NIZK proof requires a polynomially long random string (shared, but not controlled, by the prover and the verifier). So we consider the computational entropy of X, *conditioned* on a polynomially long random string r chosen from the uniform distribution $Z = U_{q(n)}$.

Let $G : \{0,1\}^n \to \{0,1\}^{\lambda(n)}$, for some polynomial λ, be a pseudorandom generator (in order to avoid adding assumptions, we can build based on Assumption 1), and let $((\mathsf{P}, \mathsf{V}), q)$ be the NIZK proof system guaranteed by Lemma 7 for the $\mathcal{NP}$ language $L = \{\phi \mid \exists \alpha \text{ such that } \phi = G(\alpha)\}$. Let $Z = R = U_{q(n)}$. Our random variable X consists of two parts $(G(U_n), \pi)$, where π is the proof, generated by P, that the first part is an output of G. More precisely, the joint distribution (X, Z) is defined as $\{\alpha \leftarrow U_n \ ; \ r \leftarrow U_{q(n)} \ ; \ \pi \leftarrow \mathsf{P}(G(\alpha), \alpha, r) : ((G(\alpha), \pi), r)\}$. Note that because X contains a proof relative to the random string r, it is defined only after the value r of Z is fixed.

Lemma 8 (Low HILL entropy). $\mathbf{H}^{\mathsf{HILL}}(X|Z) < 3n + 1$.

Proof. Suppose there is some collection $\{Y_r\}_{r \in Z}$ for which $\tilde{\mathbf{H}}_\infty(Y|Z) \geq 3n + 1$. We will show that there is a distinguisher that distinguishes (X, Z) from (Y, Z). In fact, we will use the verifier V of the NIZK proof system as a universal distinguisher, which works for every such Y.

Let $p(r) \overset{\text{def}}{=} \max_y \mathbf{Pr}[Y_r = y]$ be the probability of most likely value of the random variable Y_r.

When r is good, the number of (ϕ, π) pairs for which $\mathsf{V}(\phi, \pi, r) = 1$ is at most 2^{3n}: the total number 2^n of witnesses times the number of proofs 2^{2n} for each witness. Now, parse y as a theorem-proof pair. The number of y such that $\mathsf{V}(y, r) = 1$ is at most 2^{3n}, and each of these y happens with probability at most $p(r)$. Therefore, when r is good, $\mathbf{Pr}_{y \leftarrow Y_r}[\mathsf{V}(y, r) = 1] \leq 2^{3n} p(r)$, by the union bound. Hence, for any r, $\mathbf{Pr}_{y \leftarrow Y_r}[\mathsf{V}(y, r) = 1 \wedge r \text{ is good}] \leq 2^{3n} p(r)$ (for good r this is the same as above, and for bad r this probability is trivially 0, because of the conjunction).

Now consider running V on a sample from (Y, Z).

$$\begin{aligned}
\Pr_{(y,r)\leftarrow(Y,Z)}[\mathsf{V}(y,r)=1] &\le \Pr_{r\leftarrow Z}[r \text{ is bad}] + \Pr_{(y,r)\leftarrow(Y,Z)}[\mathsf{V}(y,r)=1 \wedge r \text{ is good}] \\
&\le \operatorname{negl}(n) + \mathop{\mathbf{E}}_{r\leftarrow Z}[\Pr_{y\leftarrow Y_r}[\mathsf{V}(y,r)=1 \wedge r \text{ is good}]] \\
&\le \operatorname{negl}(n) + \mathop{\mathbf{E}}_{r\leftarrow Z}[2^{3n}p(r)] \\
&\le \operatorname{negl}(n) + \frac{1}{2}
\end{aligned}$$

(the last inequality follows from the definition of $\tilde{\mathbf{H}}_\infty$: $2^{-\tilde{\mathbf{H}}_\infty(Y|Z)} = \mathbf{E}_{r\leftarrow Z}[p(r)] \le 2^{-(3n+1)}$).

Since $\mathbf{Pr}_{(x,r)\leftarrow(X,Z)}[\mathsf{V}(x,r)=1]=1$, V distinguishes (X,Z) from (Y,Z) with advantage close to $1/2$. □

Lemma 9 (High Yao entropy). *If Assumption 1 holds, then* $\mathbf{H}^{\mathsf{Yao}}(X|Z) \ge \lambda(n)$.

Proof. Let $s(n)$ be a polynomial. The following two statements imply that under Assumption 1, $\epsilon_s(n) \stackrel{\text{def}}{=} \mathtt{cdist}_{s(n)}((X,Z), \mathsf{SIM}(U_{\lambda(n)}))$ is negligible, by the triangle inequality.

1. $\mathtt{cdist}_{s(n)}((X,Z), \mathsf{SIM}(G(U_n)))$ is negligible. Indeed, fix a seed $\alpha \in \{0,1\}^n$ for G, and let $(X_\alpha, Z) = \{r \leftarrow U_{q(n)}; \pi \leftarrow \mathsf{P}(G(\alpha),\alpha,r) : ((G(\alpha),\pi),r)\}$. By the zero-knowledge property, we know that $\mathtt{cdist}_{s(n)}((X_\alpha,Z), \mathsf{SIM}(G(\alpha)))$ is negligible. Since it holds for every $\alpha \in \{0,1\}^n$, it also holds for a random α; we conclude that $\mathtt{cdist}_{s(n)}((X,Z), \mathsf{SIM}(G(U_n)))$ is negligible.

2. $\mathtt{cdist}_{s(n)}(\mathsf{SIM}(U_{\lambda(n)}), \mathsf{SIM}(G(U_n)))$ is negligible, because G is a pseudorandom generator.

By definition of $\epsilon_s(n)$, if the compressor and decompressor c and d have total size t, then

$$\left| \Pr_{(x,z)\leftarrow(X,Z)}[d(c(x,z),z)=x] - \Pr_{(x,z)\leftarrow\mathsf{SIM}(U_{\lambda(n)})}[d(c(x,z),z)=x] \right| \le \epsilon_s(n),$$

where $s = t +$ (size of circuit to check equality of strings of length $|x|$), because we can use $d(c(\cdot,\cdot),\cdot)$ together with the equality operator as a distinguisher.

Let the output length of c be ℓ. Then $\mathbf{Pr}_{(x,z)\leftarrow\mathsf{SIM}(U_{\lambda(n)})}[d(c(x,z),z)=x] \le 2^{\ell-\lambda(n)}$, because for every fixed z, x contains $\phi \in U_{\lambda(n)}$ (because by Lemma 7, z is independent of

ϕ in the NIZK system we use). Hence $\mathbf{Pr}_{(x,z)\leftarrow(X,Z)}[d(c(x,z),z) = x] \leq 2^{\ell-\lambda(n)} + \epsilon_s(n)$, and $\mathbf{H}^{\mathsf{Yao}}_{\epsilon_s(n),t(n)}(X|Z) \geq \lambda(n)$. For every polynomial $t(n)$, the value $s(n)$ is polynomially bounded, and therefore $\epsilon_s(n)$ is negligible, so $\mathbf{H}^{\mathsf{Yao}}(X|Z) \geq \lambda(n)$. □

Remark 1. In the previous paragraph, we could consider also the simulated proof π (recall $x = (\phi, \pi)$) when calculating $\mathbf{Pr}_{(x,z)\leftarrow\mathsf{SIM}(U_{\lambda(n)})}[d(c(x,z),z) = x]$ for even higher Yao entropy. A simulated proof π contains many random choices made by the simulator. Although the simulator algorithm for [34] is not precisely specified, but rather inferred from the simulator in [5], it is quite clear that the simulator will get to flip at least three random coins per clause in the 3-CNF formula produced out of ϕ in the reduction to 3-SAT (these three coins are needed in order to simulate the location of the $(0,0,0)$ triple [34, proof of Theorem 1, step 9] among the eight triples). This more careful calculation of $\mathbf{Pr}_{(x,z)\leftarrow\mathsf{SIM}(U_{\lambda(n)})}[d(c(x,z),z) = x]$ will yield the slightly stronger statement $\mathbf{H}^{\mathsf{Yao}}(X|Z) \geq \lambda(n) + 3\gamma(n)$, where $\gamma(n)$ is the number of clauses in the 3-CNF formula. This more careful analysis is not needed here, but will be used in Section 11.3.

Since for any polynomial $\lambda(n)$, we have pseudorandom generators of stretch λ, Lemma 8 and Lemma 9 yield the following theorem.

Theorem 10 (Separation). *Under the Quadratic Residuosity Assumption, for every polynomial λ, there exists a joint distribution ensemble $(X^{(n)}, Z^{(n)})$ such that $\mathbf{H}^{\mathsf{Yao}}(X^{(n)} \mid Z^{(n)}) \geq \lambda(n)$ and $\mathbf{H}^{\mathsf{HILL}}(X^{(n)} \mid Z^{(n)}) \leq 3n + 1$. Moreover, $Z^{(n)} = U_{q(n)}$ for some polynomial $q(n)$.*

Chapter 11

Randomness Extraction

As mentioned in the introduction, one of the main applications of computational entropy is the extraction of pseudorandom bits. Based on Theorem 10, in this section we show that the analysis based on Yao entropy can yield many more pseudorandom bits than the traditional analysis based on HILL entropy. Although Theorem 10 is for the conditional setting, we will see an example of extraction that benefits from the conditional-Yao-entropy analysis for the unconditional setting as well.

Before talking about extracting pseudorandom bits from computational entropy, let us look at a tool for analogous task in the information-theoretic setting: an *extractor* takes a distribution Y of min-entropy k, and with the help of a uniform string called the seed, "extracts" the randomness contained in Y and outputs a string of length m that is *almost uniform* even given the seed.

Definition 10 ([42]). *A polynomial-time computable function $E : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m \times \{0,1\}^d$ is a strong (k, ϵ)-extractor if the last d output bits of E are equal to the last d input bits (these bits are called* seed*), and $\mathtt{dist}((E(X, U_d), U_m \times U_d) \leq \epsilon$ for every distribution X on $\{0,1\}^n$ with $\mathbf{H}_\infty(X) \geq k$. The number of extracted bits is m, and the entropy loss is $k - m$.*

There is a long line of research on optimizing the parameters of extractors: minimizing seed length, minimizing ϵ, and maximizing m. For applications of primary interest here—using extracted randomness for cryptography—seed length is less important, because strong extractors can use non-secret random seeds, which are usually much easier to create than the secret from which the pseudorandom bits are being extracted. It is more important to maximize m (as close to k as possible), while keeping ϵ negligible.[1]

11.1 Extracting from Conditional HILL Entropy

It is not hard to see that applying an extractor on distributions with HILL entropy yields pseudorandom bits; because otherwise the extractor together with the distinguisher violate the definition of HILL entropy. We show the same for the case of conditional HILL entropy. We reiterate that in the conditional case, the variable Z is given to the distinguisher who is trying to tell the output of the extractor from random.

Lemma 11. *If* $\mathbf{H}^{\mathsf{HILL}}_{\epsilon_1,s}(X|Z) \geq k$, *then for any* $(k - \log\frac{1}{\delta}, \epsilon_2)$*-extractor* $E : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$,

$$\underset{s'}{\mathtt{cdist}}\left(\{(x,z) \leftarrow (X,Z) : (E(x,U_d),z)\}, U_m \times U_d \times Z\right) \leq \epsilon_1 + \epsilon_2 + \delta\,,$$

where $s' = s - \mathit{size}(E)$.

Proof. $\mathbf{H}^{\mathsf{HILL}}_{\epsilon_1,s}(X|Z) \geq k$ means that there is a collection of distributions $\{Y_z\}_{z\in Z}$ such that $\mathtt{cdist}_s((X,Z)(Y,Z)) \leq \epsilon_1$, and $\tilde{\mathbf{H}}_\infty(Y|Z) \geq k$. By Markov's inequality, $\mathbf{Pr}_{z\in Z}[\mathbf{H}_\infty(Y_z) \leq k - \log\frac{1}{\delta}] \leq \delta$. Hence, the extractor works as expected in all but δ fraction of the cases; that is, for all but δ fraction of z values, $\mathtt{dist}(E(Y_z,U_d), U_m \times U_d) \leq \epsilon_2$. Taking expectation over $z \in Z$, we get

$$\mathtt{dist}\left(\{(y,z) \leftarrow (Y,Z) : (E(y,U_d),z)\}, U_m \times U_d \times Z\right) \leq \epsilon_2 + \delta\,,$$

because $\mathtt{dist}$ is bounded by 1. The desired result follows by triangle inequality. □

Remark 2. The entropy loss of E is at least $2\log\frac{1}{\epsilon_2} - O(1)$, by a fundamental constraint on extractors [47], giving us a total entropy loss of at least $\log\frac{1}{\delta} + 2\log\frac{1}{\epsilon_2} - O(1)$. The loss of $\log\frac{1}{\delta}$ can be avoided for some specific E, such as pairwise-independent (a.k.a. strongly

[1] This is in contrast to the derandomization literature, where a small constant ϵ suffices, and one is more interested in (simultaneously) maximizing m and minimizing d.

universal) hashing [8], as shown in [14, Lemma 4.2]; because pairwise-independent hashing has optimal entropy loss of $2\log\frac{1}{\epsilon_2} - 2$, this gives us the maximum possible number of extracted bits. The loss of $\log\frac{1}{\delta}$ can be also avoided when $\min_{z\in Z}\mathbf{H}_\infty(Y_z) \geq k$ (as is the case in, e.g., [15]).

Using an extractor on distributions with HILL entropy (the method that we just showed extends to conditional HILL entropy) is a common method for extracting pseudorandom bits. HILL entropy is used, in particular, because it is easier to analyze than Yao entropy. In fact, in the unconditional setting, the only way we know how to show that a distribution has high Yao entropy (incompressibility) is by arguing that it has high HILL entropy (indistinguishability). Nevertheless, Barak et al. [3] showed that some extractors can also extract from Yao entropy.

11.2 Extracting from Conditional Yao Entropy

Barak et al. [3] observed that extractors with the so-called reconstruction procedure can be used to extract from Yao Entropy. Thus, Theorem 10 ($\mathbf{H}^{\mathsf{Yao}}(X|Z) \gg \mathbf{H}^{\mathsf{HILL}}(X|Z)$) suggests that such a *reconstructive* extractor with a Yao-entropy-based analysis may yield more pseudorandom bits than a generic extractor with a traditional HILL-entropy-based analysis. We begin with a definition from [3], with minor modifications for our purposes (see explanation following the definition).

Definition 11 (Reconstruction procedure). *An (ℓ, ϵ)-reconstruction for a function $E : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m \times \{0,1\}^d$ (where the last d output bits are equal to the last d input bits) is a pair of randomized polynomial-size oracle circuits $C^{(\cdot)} : \{0,1\}^n \to \{0,1\}^\ell$ and $D^{(\cdot)} : \{0,1\}^\ell \to \{0,1\}^n$, that share the same random coins. Furthermore, for every x and T, if $|\mathbf{Pr}[T(E(x,U_d)) = 1] - \mathbf{Pr}[T(U_m \times U_d) = 1]| > \epsilon$, then $\mathbf{Pr}[D^T(C^T(x)) = x] > 1/2$ (the probability is over the shared random coins of C and D).*

Our definition of reconstruction procedure differs from [3] in the following ways:

- It is with respect to *strong* extractors[2] (as opposed to generic extractors) since we will only be using extractors of this kind.
- We let C and D be circuits instead of Turing machines because we define the Yao entropy with respect to circuit adversaries.
- We let C and D share their random coins. The advantage is that ℓ can be smaller because the output of C does not need to contain anything about the coins. This is especially useful when later we want to extract almost all of Yao entropy from a distribution, using a reconstructive extractor. We also give C oracle access to T. This broadens the class of possible C and D (since if C doesn't use its oracle and uses a subset of random coins that are not used by D, then we are in the original definition of [3]), and makes Lemma 12 applicable to more situations. It also enables us to prove Theorem 14 later.

Trevisan [58] showed, implicitly, that any E with an (ℓ, ϵ)-reconstruction is an $(\ell + \log \frac{1}{\epsilon}, 3\epsilon)$-extractor. Thus, any function with a reconstruction procedure is an extractor; it is called a *reconstructive* extractor.

Barak et al. [3] showed that reconstructive extractors can be used to extract pseudorandom bits from distributions with Yao entropy. We extend the proof of Barak et al. so that their result holds for the conditional version of Yao entropy.

Lemma 12. *Let X be a distribution with $\mathbf{H}^{\mathsf{Yao}}_{\epsilon,s}(X|Z) \geq k$, and let E be an extractor with an (ℓ, ϵ)-reconstruction (C, D), where $\ell = k - \log \frac{1}{\epsilon}$. Then E extracts psuedorandom bits from X: namely, $\texttt{cdist}_{s'}((E(X, U_d), Z), U_m \times U_d \times Z) \leq 5\epsilon$, where $s' = s/(size(C)+size(D))$.*

Proof. Assume, for the purpose of contradiction, that there is a distinguisher T of size s' such that $|\mathbf{Pr}[T(E(X, U_d), Z) = 1] - \mathbf{Pr}[T(U_m \times U_d \times Z) = 1]| > 5\epsilon$. By the Markov inequality, there is a subset S in the support of (X, Z) such that $\mathbf{Pr}[(X, Z) \in S] \geq 4\epsilon$, and $\forall (x, z) \in S$, $|\mathbf{Pr}[T(E(x, U_d), z) = 1] - \mathbf{Pr}[T(U_m \times U_d, z) = 1]| > \epsilon$. For every pair $(x, z) \in S$,

[2]Recall that a strong extractor is an extractor that also outputs its seed.

$\mathbf{Pr}[D^{T(\cdot,z)}(C^{T(\cdot,z)}(x)) = x] > 1/2$, where the probability is over the shared random coins of C and D (note that fixing z is important, because C and D are defined with respect to an oracle that does not expect z as an input). Thus, there is a fixing of the random choices of C and D, denoted by circuits $\bar{C}, \bar{D}$, such that $\mathbf{Pr}_{(x,z)\leftarrow(X,Z)}[\bar{D}^{T(\cdot,z)}(\bar{C}^{T(\cdot,z)}(x)) = x] > 2\epsilon$. Let $c(x,z) = \bar{C}^{T(\cdot,z)}(x)$ and $d(y,z) = \bar{D}^{T(\cdot,z)}(y)$ be the compression and decompression circuits, respectively. Then $\mathbf{Pr}_{(x,z)\leftarrow(X,Z)}[d(c(x,z),z) = x] > 2\epsilon = 2^{\ell-k} + \epsilon$, a contradiction. □

The above lemma does not yield more pseudorandom bits when given a distribution that has high Yao but low HILL entropy, unless we have a reconstructive extractor with long output length (compared to generic extractors, which work for HILL entropy). Fortunately, given any reconstructive extractor, there is a simple way to increase the number of pseudorandom bits extracted: apply the extractor multiple times on the same input distribution but each time with an independent fresh seed.[3] Furthermore, there do exist reconstructive extractors; e.g., the well-known Goldreich-Levin extractor: $GL(x,r) \stackrel{\text{def}}{=} (x \cdot r) \circ r$, where $\circ$ denotes concatenation and $\cdot$ denotes inner product, is a reconstructive one. Below, we describe more precisely how to extract most of Yao entropy from a distribution using the GL extractor.

We first rephrase a lemma from [19] to show that GL is a reconstructive extractor, and then show that we can indeed apply GL on the same input distribution multiple times.

Lemma 13 (Goldreich-Levin [19]). *There is a randomized oracle Turing machine I such that given any $\epsilon > 0$ and any oracle $T : \{0,1\}^{n+1} \to \{0,1\}$, runs in time $poly(n, \frac{1}{\epsilon})$ and outputs a set L of size $poly(n, \frac{1}{\epsilon})$ so that for every $x \in \{0,1\}^n$*

$$\left| \mathop{\mathbf{Pr}}_{r\leftarrow U_n}[T(x\cdot r, r) = 1] - \mathop{\mathbf{Pr}}_{u\leftarrow U_1, r\leftarrow U_n}[T(u,r) = 1] \right| > \epsilon \quad \Rightarrow \quad \mathop{\mathbf{Pr}}_{I\text{'s coins}}[x \in L] > \frac{1}{2}$$

For a proof, see Lemma 1 in [19] and Theorem 1 in [57].

[3]This requires a long seed, but as mentioned earlier in this Chapter, our primary concern is on the extracted randomness instead of seed length.

To see that GL is reconstructive, simply let the reconstruction procedure (C, D) run I to get the set L. On input x, the machine C outputs the *index* of x in the set L (or some arbitrary string if $x \notin L$) so that later D can restore x. Since C and D share the same random coins, they produce the same L. And x can be reconstructed with probability more than 1/2.

Theorem 14. *Let $GL : \{0,1\}^n \times \{0,1\}^n \to \{0,1\} \times \{0,1\}^n$ be the Goldreich-Levin extractor with (ℓ, ϵ)-reconstruction (C, D), and let X be a distribution over $\{0,1\}^n$ with $\mathbf{H}^{\mathsf{Yao}}_{\epsilon,s}(X|Z) \geq \rho + \ell + \log\frac{1}{\epsilon}$. Define $E : \{0,1\}^n \times \{0,1\}^{\rho n} \to \{0,1\}^{\rho} \times \{0,1\}^{\rho n}$ as follows:*

$$E(x, r_1, \ldots, r_\rho) \overset{\text{def}}{=} ((x \cdot r_1) \circ \cdots \circ (x \cdot r_\rho)) \circ (r_1 \circ \cdots \circ r_\rho).$$

Then $\mathtt{cdist}_{s'}((E(X, U_{\rho n}), Z), U_\rho \times U_{\rho n} \times Z) \leq 5\rho\epsilon$, where $s' = \frac{s - O(\rho n)}{size(C) + size(D)}$.

Proof. The proof is by hybrid argument. For a fixed x and z, let $p(j)$ be $\mathbf{Pr}[T(((x \cdot r_1) \circ \cdots \circ (x \cdot r_j) \circ b_{j+1} \circ \cdots \circ b_\rho) \circ (r_1 \circ \cdots \circ r_\rho), z) = 1]$, where the probability is over $\{r_1, \ldots, r_\rho \leftarrow U_n, b_{j+1}, \ldots, b_\rho \leftarrow U_1\}$. Now assume, for the purpose of contradiction, that there is a distinguisher T of size s' such that $|\mathbf{Pr}[T(E(X, U_{\rho n}), Z) = 1] - \mathbf{Pr}[T(U_\rho \times U_{\rho n} \times Z) = 1]| > 5\rho\epsilon$; that is, $|\mathbf{E}_{(x,z)\leftarrow(X,Z)}[p(\rho) - p(0)]| > 5\rho\epsilon$. By the triangle inequality, there exists an $i \in \{1, \ldots, \rho\}$ such that $|\mathbf{E}_{(x,z)\leftarrow(X,Z)}[p(i) - p(i-1)]| > 5\epsilon$. So, there exists a fixing of all the r_j and b_j except r_i and b_i for which $|\mathbf{E}_{(x,z)\leftarrow(X,Z)}[p_1 - p_0]| > 5\epsilon$, where

$$p_1 = \mathbf{Pr}[T(((x \cdot r_1) \circ \cdots \circ (x \cdot r_{i-1}) \circ \quad (x \cdot r_i) \quad \circ b_{i+1} \circ \cdots \circ b_\rho) \circ (r_1 \circ \cdots \circ r_\rho), z) = 1]$$
$$p_0 = \mathbf{Pr}[T(((x \cdot r_1) \circ \cdots \circ (x \cdot r_{i-1}) \circ \quad b_i \quad \circ b_{i+1} \circ \cdots \circ b_\rho) \circ (r_1 \circ \cdots \circ r_\rho), z) = 1]$$

and the probabilities are taken over only r_i and b_i.By the Markov inequality, there is a subset S in the support of (X, Z) such that $\mathbf{Pr}[(X, Z) \in S] \geq 4\epsilon$, and $\forall (x, z) \in S$, $|p_1 - p_0| > \epsilon$.

In order to get a contradiction, we want to use C and D to compress x given z. C and D need to call a distinguisher that can distinguish $((x \cdot r_i), r_i)$ from (b_i, r_i). However, we do not have such a distinguisher: rather, we have T, which, from the above analysis, we know we can use for some fixed set of r_j and b_j (for $j \neq i$) as long as we also provide it with $x \cdot r_j$

for $j < i$. The fixed set of r_j and b_j for $j \neq i$ can be simply hardwired into C and D. The values $x \cdot r_j$ are more problematic: C can compute them because it has access to x, but D cannot. So we need to modify C to compute the $x \cdot r_j$ and append them to its output, and modify D to use them. Call the resulting modified procedures $(\bar{C}, \bar{D})$. Note that the output length of $\bar{C}$ is at most $\rho + \ell - 1$, because it appends at most $\rho - 1$ bits to the output of C.

For every pair $(x, z) \in S$, $\mathbf{Pr}[\bar{D}^{T(\cdot,z)}(\bar{C}^{T(\cdot,z)}(x)) = x] > 1/2$, where the probability is over the shared random coins of $\bar{C}$ and $\bar{D}$. Thus, there is a fixing of the shared random coins of $\bar{C}$ and $\bar{D}$, denoted by circuits $\hat{C}, \hat{D}$, such that $\mathbf{Pr}_{(x,z)\leftarrow(X,Z)}[\hat{D}^{T(\cdot,z)}(\hat{C}^{T(\cdot,z)}(x)) = x] > 2\epsilon$. Let $c(x, z) = \hat{C}^{T(\cdot,z)}(x)$ and $d(y, z) = \hat{D}^{T(\cdot,z)}(y)$ be the compression and decompression circuits, respectively. Recall that the output length of c is less than $\rho + \ell$, and thus $\mathbf{Pr}_{(x,z)\leftarrow(X,Z)}[d(c(x, z), z) = x] > 2\epsilon = 2^{(\rho+\ell)-(\rho+\ell+\log\frac{1}{\epsilon})} + \epsilon$, a contradiction. $\square$

For the Goldreich-Levin extractor, ℓ is the length of the index for the set L, which is $O(\log \frac{n}{\epsilon})$. Then Theorem 14 shows that E extracts ρ pseudorandom bits out of any distribution that has Yao entropy $\rho + \ell + \log \frac{1}{\epsilon} = \rho + O(\log \frac{n}{\epsilon})$, which means that it is possible to extract almost all of Yao entropy (e.g., if the negligible $\epsilon = 2^{-\mathrm{polylog}(n)}$ suffices, then all but a polylogarithmic amount of entropy can be extracted).

We remark, however, that the resulting extractor is *not* a reconstructive one. The reason being that by our definition, reconstruction procedures C and D do *not* depend on T. This property of C and D is crucial to the proof of Lemma 12 because the distinguisher T depends on z. If C and D were to depend on T, they would depend on z too, which would make the proof fail. This is in contrast to the unconditional setting, where we may allow reconstruction procedures to depend on T (see, e.g., Section 3.3 in the survey by Shatiel [53]: there exist C_T and D_T for every T) and still be able to extract pseudorandomness from Yao entropy.

Theorem 14 can be generalized to strong extractors (not just Goldreich-Levin) with output length $m+d$, for $m > 1$. More precisely, we can extract ρm (instead of ρ) pseudorandom bits out of any distribution that has Yao entropy $\rho m + \ell + \log \frac{1}{\epsilon}$ (instead of $\rho + \ell + \log \frac{1}{\epsilon}$).

Using the distribution of Theorem 10, we can set $\epsilon = 2^{-n}$ to extract $\lambda(n) - O(n)$ bits from X that are pseudorandom even given Z (note that here n is not the length of X; rather, $\lambda(n)$ is). This is more than the linear number of bits extractable from X using the analysis based on conditional HILL entropy.

11.3 Unconditional Extraction

In this section, let $(X, Z) = ((G(U_n), \Pi), R) = \{\alpha \leftarrow U_n \ ; \ r \leftarrow U_{q(n)} \ ; \ \pi \leftarrow \mathcal{P}(G(\alpha), \alpha, r) : ((G(\alpha), \pi), r)\}$ as defined in Section 10.3. The question is: how many pseudorandom bits can we extract from the unconditional distribution (X, Z)? Surprisingly, analysis based on conditional entropy yields more bits than unconditional analysis, demonstrating that the notion of conditional entropy may be a useful tool even in the analysis of pseudorandomness of unconditional distributions.

Analysis based on unconditional entropy. The straightforward way is to apply an extractor on (X, Z). This gives us almost k pseudorandom bits provided that $\mathbf{H}^{\mathsf{HILL}}(X, Z) \geq k$, or $\mathbf{H}^{\mathsf{Yao}}(X, Z) \geq k$ for reconstructive extractors (see previous sections). However, the best we can show is that $\mathbf{H}^{\mathsf{HILL}}(X, Z) = \lambda(n) + q(n) + O(n)$ (the analysis appears in Appendix B), and hence we cannot prove, using HILL entropy, that more than $\lambda(n) + q(n) + O(n)$ bits can be extracted. On the other hand, we do not know if $\mathbf{H}^{\mathsf{Yao}}(X, Z)$ is higher; this is closely related to the open problem of whether HILL entropy is equivalent to Yao entropy, and appears to be difficult.[4] Thus, analysis based on unconditional entropy does not seem to yield more than $\lambda(n) + q(n) + O(n)$ bits.

[4] To show that $\mathbf{H}^{\mathsf{Yao}}(X, Z)$ is high, one would have to show that the pair (X, Z) cannot be compressed; the same indistinguishability argument as in Lemma 9 does not work for the pair (X, Z), because in the simulated distribution, Z is simulated and thus has less entropy. It is thus possible that both the real distribution (where Z is random and ϕ in X is pseudorandom) and the simulated distribution (where ϕ is random and Z is pseudorandom), although indistinguishable, can be compressed with the help of the proof π.

More bits from conditional Yao entropy. Analysis based on conditional HILL entropy seems to yield even fewer bits (see Lemma 8). But using conditional Yao entropy, we get the following result.

Lemma 15. *It is possible to extract* $4\lambda(n) + q(n) - O(n)$ *pseudorandom bits out of* (X, Z).

Sketch. According to Remark 1 following Lemma 9, we can show that the conditional Yao entropy $\mathbf{H}^{\mathsf{Yao}}(X|Z) \geq \lambda(n) + 3\gamma(n)$, where $\gamma(n)$ is the number of clauses in the 3-CNF formula produced from ϕ in the reduction from L to 3-SAT. Since $\gamma(n) \geq \lambda(n)$, we can extract $4\lambda(n) - O(n)$ bits from X that are pseudorandom even given Z, by the last paragraph of Section 11.2. Noting that Z is simply a uniform string[5], we can append it to the pseudorandom bits extracted from X and obtain an even longer pseudorandom string. Thus, we get $4\lambda(n) + q(n) - O(n)$ pseudorandom bits using the analysis based on conditional Yao entropy. □

[5]In case Z is not uniform but contains some amount of entropy, we can apply another extractor on it.

Chapter 12

Unpredictability Entropy

In this chapter, we introduce a new computational entropy, which we call unpredictability entropy. Analogous to min-entropy, which is the logarithm of the maximum predicting probability, unpredictability entropy is the logarithm of the maximum predicting probability where the predictor is restricted to be a circuit of polynomial size. Note that in the unconditional setting, unpredictability entropy is just min-entropy; a small circuit can have the most likely value hardwired. In the conditional setting, however, this new definition can be very different from min-entropy, and in particular, allows us to talk about the entropy of a value that is unique, such as g^{xy} where g^x and g^y are known to the observer, and possibly even verifiable, such as the preimage x of a one-way permutation f, where $y = f(x)$ is known to the observer.

Definition 12 (Unpredictability entropy). *For a distribution (X, Z), we say that X has* **unpredictability entropy** *at least k conditioned on Z, denoted by $\mathbf{H}^{\mathsf{unp}}_{\epsilon,s}(X|Z) \geq k$, if there exists a collection of distributions Y_z (giving rise to a joint distribution (Y, Z)) such that $\mathtt{cdist}_s((X,Z),(Y,Z)) \leq \epsilon$, and for all circuits C of size s,*

$$\mathbf{Pr}[C(Z) = Y] \leq 2^{-k}.$$

Remark 3. The parameter ϵ and the variable Y do not seem to be necessary in the definition; we can simply require $\mathbf{Pr}[C(Z) = X] \leq 2^{-k}$. However, they make this definition *smooth* [49] and easier to compare with existing definitions of HILL and Yao entropy.

Remark 4. Note that our entropy depends primarily on the predicting probability, as opposed to on the size of the predicting circuit or the combination of both (see e.g., [56, 25]). We choose to have s fixed, in order to accommodate distributions with nonzero information-theoretic entropy; otherwise the computational entropy of such distribution would be infinite because the predicting probability doesn't increase no matter how big the predicting circuit grows. For the case of one-way function, unpredictability entropy is what is often called "hardness." This notion is more general, and provides a simple language for pseudorandomness extraction: namely, a distribution with computational entropy k contains k pseudorandom bits that can be extracted (see below).

Relation to Other Notions and Bit Extraction In the rest of this chapter we show that high conditional HILL entropy implies high unpredictability entropy, which in turn implies high conditional Yao entropy. Note that, assuming exponentially strong one-way permutations f exist, unpredictability entropy does not imply conditional HILL entropy: simply let $(X, Z) = (x, f(x))$.

Lemma 16. $\mathbf{H}^{\mathsf{HILL}}_{\epsilon,s}(X|Z) \geq k \Rightarrow \mathbf{H}^{\mathsf{unp}}_{\epsilon,s}(X|Z) \geq k$.

Proof. $\mathbf{H}^{\mathsf{HILL}}_{\epsilon,s}(X|Z) \geq k$ means that there is a collection of $\{Y_z\}_{z\in Z}$ such that $\tilde{\mathbf{H}}_\infty(Y|Z) \geq k$ and $\mathtt{cdist}_s((X,Z),(Y,Z)) \leq \epsilon$. And $\tilde{\mathbf{H}}_\infty(Y|Z) \geq k$ means that $\mathbf{E}_{z\leftarrow Z}[\max_y \mathbf{Pr}[Y = y|Z = z]] \leq 2^{-k}$, which implies that for all circuits C of size s, $\mathbf{Pr}[C(Z) = Y] \leq 2^{-k}$. □

Lemma 17. $\mathbf{H}^{\mathsf{unp}}_{\epsilon,s}(X|Z) \geq k \Rightarrow \mathbf{H}^{\mathsf{Yao}}_{\epsilon,s}(X|Z) \geq k$.

Proof. $\mathbf{H}^{\mathsf{unp}}_{\epsilon,s}(X|Z) \geq k$ means that there is a collection of distributions $\{Y_z\}_{z\in Z}$ such that $\mathtt{cdist}_s((X,Z),(Y,Z)) \leq \epsilon$, and for all circuits C of size s, $\mathbf{Pr}[C(Z) = Y] \leq 2^{-k}$. We will show that $\mathbf{H}^{\mathsf{Yao}}_{0,s}(Y|Z) \geq k$, which in turn implies $\mathbf{H}^{\mathsf{Yao}}_{\epsilon,s}(X|Z) \geq k$.

Suppose for contradiction that $\mathbf{H}^{\mathsf{Yao}}_{0,s}(Y|Z) < k$. Then there exists a pair of circuits c, d of total size s with the outputs of c having length ℓ, such that $\mathbf{Pr}_{(y,z)\leftarrow(Y,Z)}[d(c(y,z),z) = y] > 2^{\ell-k}$. Because $|c(y,z)| = \ell$, guessing the correct value of $c(y,z)$ is at least $2^{-\ell}$, so

$\mathbf{Pr}_{(a,y,z)\leftarrow(U_\ell,Y,Z)}[d(a,z) = y] > 2^{\ell-k} \cdot 2^{-\ell} = 2^{-k}$, a contradiction since $d(a,\cdot)$ (with some fixing of a) is a circuit of size at most s. So $\mathbf{H}^{\mathsf{Yao}}_{0,s}(Y|Z) \geq k$.

Next, suppose for contradiction that $\mathbf{H}^{\mathsf{Yao}}_{\epsilon,s}(X|Z) < k$. Then there is a pair of circuits c, d of total size s with the outputs of c having length ℓ, such that $\mathbf{Pr}_{(x,z)\leftarrow(X,Z)}[d(c(x,z),z) = x] > 2^{\ell-k} + \epsilon$. But $\mathbf{Pr}_{(y,z)\leftarrow(Y,Z)}[d(c(y,z),z) = y] \leq 2^{\ell-k}$, which means that $d(c(\cdot,\cdot),\cdot)$ can be used to distinguish (X,Z) from (Y,Z) with advantage more than ϵ, a contradiction to $\mathtt{cdist}_s((X,Z),(Y,Z)) \leq \epsilon$. Hence $\mathbf{H}^{\mathsf{Yao}}_{\epsilon,s}(X|Z) \geq k$. □

From Section 11.2, we know how to extract almost k bits from distributions with Yao entropy k, by using reconstructive extractors. Lemma 17 implies that the same method works for unpredictability entropy. Thus, the notion of unpredictability entropy allows for more general statements of results on hardcore bits (such as, for example, [19, 56]), which are usually formulated in terms of one-way functions. Most often these results generalize easily to other conditionally unpredictable distributions, for instance, the Diffie-Hellman distribution $(g^{xy} \mid g, g^x, g^y)$. However, such generalization is not automatic, because a prediction of a one-way function inverse is verifiable (namely, knowing y, one can check if the guess for $f^{-1}(y)$ is correct), while a guess of a value of a conditionally unpredictable distribution in general is not (indeed, the Diffie-Hellman distribution does not have it unless the decisional Diffie-Hellman problem is easy). Thus, it would be beneficial if results were stated for the more general case of unpredictable distributions whenever such verifiability is not crucial. Unpredictability entropy provides a simple language for doing so.

Appendix A

Modifications to the Proof of [34]

The proof of Theorem 1 in [34] requires the n-bit modulus x chosen by the prover (and, in our case, included as part of the proof) to be a Blum integer, i.e., a product of two primes that are each congruent to 3 modulo 4. However, the proof π (using the techniques from [5]) guarantees only that x is "Regular(2)," i.e., is square-free and has exactly two distinct odd prime divisors. In other words, we are assured only that x is of the form $p^i q^j$ for some odd primes p, q and some i, j not simultaneously even. Soundness does not suffer if a prover maliciously chooses such an x that is not a Blum integer, but the uniqueness property does: there may be more than one valid proof π, because π consists of square roots s of values in $\mathbb{Z}_x^*$ such that the Jacobi symbol $\left(\frac{s}{x}\right) = 1$ and $s < x/2$, and there may be more than one such square root if x is not a Blum integer.

One approach to remedy this problem is to use the technique proposed in countable zero-knowledge of Naor [39, Theorem 4.1]: to include into π the proof that x is a Blum integer. Another, simpler, approach (which does not seem to work for the problem in [39], because the length of the primes is important there) is to require the verifier to check that $x \equiv 1 \pmod 4$. This guarantees that either $p \equiv q \equiv 3 \bmod 4$ and i, j are odd, in which case uniqueness of a square root $r < x/2$ with $\left(\frac{r}{x}\right) = 1$ is guaranteed, or $p^i \equiv q^j \equiv 1 \bmod 4$, in which case simple number theory (case analysis by the parity of i, j) shows that half the quadratic residues in $\mathbb{Z}_x^*$ have *no* square root r with $\left(\frac{r}{x}\right) = 1$. Thus, such an x that allows for non-unique proofs is very unlikely to work for a shared random string r, and we can simply add strings r for which such an x exists to the set of bad strings (which will remain of negligible size).

Appendix B

Unconditional HILL Entropy of (X, Z)

Recall that $(X,Z) = ((G(U_n),\Pi),R) = \{\alpha \leftarrow U_n \; ; \; r \leftarrow U_{q(n)} \; ; \; \pi \leftarrow \mathcal{P}(G(\alpha),\alpha,r) : ((G(\alpha),\pi),r)\}$. Below, we show that $\mathbf{H}^{\mathsf{HILL}}(X,Z) \geq \lambda(n) + q(n) + O(n)$; it is unclear if higher HILL entropy can be shown. The discussion assumes some familiarity with the NIZK system for 3-SAT, by Lepinski, Micali, and shelat [34].

By the zero-knowledgeness, the output distribution $(X_{\mathsf{SIM}}, Z_{\mathsf{SIM}})$ of the simulator is indistinguishable from (X,Z). So $\mathbf{H}^{\mathsf{HILL}}(X,Z)$ is no less than the min-entropy of $(X_{\mathsf{SIM}}, Z_{\mathsf{SIM}})$. We count how many choices the simulator SIM has: there are,

- $2^{\lambda(n)}$ theorems to prove,
- fewer than 2^{2n} proving pairs to choose from (a proving pair is an n-bit Blum integer x and an n-bit quadratic residue $y \in \mathbb{Z}_x^*$),
- $2^{q(n)-\kappa(n)}$ choices for shared "random" string r, where $\kappa(n)$ is the number of Jacobi symbol 1 elements of $\mathbb{Z}_x^*$ included in r (because in the simulated r, these elements must be quadratic residues in $\mathbb{Z}_x^*$),
- $2^{\kappa(n)}$ choices for claiming, in the simulated proof, whether each of the Jacobi symbol 1 elements in r is a quadratic residue or a quadratic nonresidue (the simulator gets to make false claims about that, because in the simulated r, they are all residues).

Taking the logarithm of the number of choices, we have $\mathbf{H}^{\mathsf{HILL}}(X, Z) \geq \lambda(n) + q(n) + O(n)$. This seems to be the best we can do, as we do not know whether there are other distribution that is indistinguishable from (X, Z).

Bibliography

[1] *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, Seattle, Washington, 15–17 May 1989.

[2] Malcolm Adams and Victor Guillemin. *Measure Theory and Probability.* Springer-Verlag, 1996.

[3] Boaz Barak, Ronen Shaltiel, and Avi Wigderson. Computational analogues of entropy. In Sanjeev Arora, Klaus Jansen, José D. P. Rolim, and Amit Sahai, editors, *RANDOM-APPROX 2003*, volume 2764 of *Lecture Notes in Computer Science*, pages 200–215. Springer, 2003.

[4] Mihir Bellare and Phillip Rogaway. Collision-resistant hashing: Towards making UOWHFs practical. In Burton S. Kaliski, Jr., editor, *Advances in Cryptology—CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 470–484. Springer-Verlag, 1997.

[5] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM Journal on Computing*, 20(6):1084–1118, December 1991.

[6] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 103–112, Chicago, Illinois, 2–4 May 1988.

[7] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–863, November 1984.

[8] J. Larry Carter and Mark N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18:143–154, 1979.

[9] Yan-Cheng Chang, Chun-Yun Hsiao, and Chi-Jen Lu. On the impossibilities of basing one-way permutations on central cryptographic primitives. In Yuliang Zheng, editor, *Advances in Cryptology—ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 110–124, Queenstown, New Zealand, 1–5 December 2002. Springer-Verlag.

[10] Ivan Damgård. Collision-free hash functions and public-key signature schemes. In David Chaum and Wyn L. Price, editors, *Advances in Cryptology—EUROCRYPT 87*, volume 304 of *Lecture Notes in Computer Science*. Springer-Verlag, 1988, 13–15 April 1987.

[11] Alfredo De Santis and Giuseppe Persiano. Zero-knowledge proofs of knowledge without interaction. In *33rd Annual Symposium on Foundations of Computer Science*, pages 427–436, Pittsburgh, Pennsylvania, 24–27 October 1992. IEEE.

[12] Nenad Dedić, Danny Harnik, and Leonid Reyzin. Saving private randomness in one-way functions and pseudorandom generators. In Ran Canetti, editor, *Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 607–625. Springer, 2008.

[13] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[14] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. Technical Report 2003/235, Cryptology ePrint archive, `http://eprint.iacr.org`, 2006. Previous version appeared at *EUROCRYPT 2004*.

[15] Rosario Gennaro, Hugo Krawczyk, and Tal Rabin. Secure hashed Diffie-Hellman over non-DDH groups. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology—EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 361–381. Springer-Verlag, 2004.

[16] Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *41st Annual Symposium on Foundations of Computer Science* [29].

[17] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *41st Annual Symposium on Foundations of Computer Science* [29], pages 325–335.

[18] Yael Gertner, Tal Malkin, and Omer Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *42nd Annual Symposium on Foundations of Computer Science*, Las Vegas, Nevada, October 2001. IEEE.

[19] Oded Goldreich and Leonid Levin. A hard-core predicate for all one-way functions. In ACM [1], pages 25–32.

[20] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. Available From `http://www.mit.edu/~tauman/`.

[21] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th Annual Symposium on Foundations of Computer Science* [30], pages 102–113.

[22] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984.

[23] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[24] Iftach Haitner. Implementing oblivious transfer using collection of dense trapdoor permutations. In Naor [40], pages 394–409.

[25] Johan Håstad, Russell Impagliazzo, Leonid Levin, and Michael Luby. Construction of pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[26] Thomas Holenstein. Pseudorandom generators from one-way functions: A simple construction for any hardness. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2006.

[27] Chun-Yuan Hsiao, Chi-Jen Lu, and Leonid Reyzin. Conditional computational entropy, or toward separating pseudoentropy from compressibility. In Moni Naor, editor, *Advances in Cryptology—EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 169–186. Springer-Verlag, 2007.

[28] Chun-Yuan Hsiao and Leonid Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins. In Matt Franklin, editor, *Advances in Cryptology—CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 92–105. Springer-Verlag, 15–19 August 2004.

[29] IEEE. *41st Annual Symposium on Foundations of Computer Science*, Redondo Beach, California, November 2000.

[30] IEEE. *44th Annual Symposium on Foundations of Computer Science*, Cambridge, Massachusetts, October 2003.

[31] Russell Impagliazzo. Remarks in open problem session at the dimacs workshop on pseudorandomness and explicit combinatorial constructions, 1999.

[32] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In ACM [1], pages 44–61.

[33] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Sufficient conditions for collision-resistant hashing. In *The 2nd Theory of Cryptography Conference*, pages 445–456, 2005.

[34] Matt Lepinski, Silvio Micali, and Abhi Shelat. Fair-zero knowledge. In Joe Kilian, editor, *Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 245–263. Springer-Verlag, 2005.

[35] Ralph C. Merkle. *Secrecy, Authentication, and Public Key Systems.* UMI Research Press, 1982.

[36] Ralph C. Merkle. A certified digital signature. In G. Brassard, editor, *Advances in Cryptology—CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238. Springer-Verlag, 1990, 20–24 August 1989.

[37] Silvio Micali, Michael Rabin, and Joe Kilian. Zero-knowledge sets. In *44th Annual Symposium on Foundations of Computer Science* [30], pages 80–91.

[38] Ilya Mironov. Hash functions: From Merkle-Damgård to Shoup. In Joe Kilian, editor, *Advances in Cryptology—CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 166–181. Springer-Verlag, 2001.

[39] Moni Naor. Evaluation may be easier than generation. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 74–83, Philadelphia, Pennsylvania, 22–24 May 1996.

[40] Moni Naor, editor. *First Theory of Cryptography Conference — TCC 2004*, volume 2951 of *Lecture Notes in Computer Science.* Springer-Verlag, February 19–21 2004.

[41] FIPS publication 180-1: Secure hash standard, April 1995. Available from `http://csrc.nist.gov/fips/`.

[42] Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–53, 1996.

[43] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *Advances in Cryptology—CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer-Verlag, 1992, 11–15 August 1991.

[44] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *The 40th ACM Symposium on Theory of Computing*, pages 187–196, 2008.

[45] Michael O. Rabin. Digitalized signatures. In Richard A. Demillo, David P. Dobkin, Anita K. Jones, and Richard J. Lipton, editors, *Foundations of Secure Computation*, pages 155–168. Academic Press, 1978.

[46] Michael O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, Massachusetts Institute of Technology, Cambridge, MA, January 1979.

[47] Jaikumar Radhakrishnan and Amnon Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM Journal on Computing*, 13(1):2–24, 2000.

[48] Omer Reingold, Luca Trevisan, and Salil Vadhan. Notions of reducibility between cryptographic primitives. In Naor [40], pages 1–20.

[49] Renato Renner and Stefan Wolf. Smooth rényi entropy and applications. In *Proceedings of IEEE International Symposium on Information Theory*, page 233, June 2004.

[50] Renato Renner and Stefan Wolf. Simple and tight bounds for information reconciliation and privacy amplification. In Bimal Roy, editor, *Advances in Cryptology—ASIACRYPT 2005*, Lecture Notes in Computer Science, Chennai, India, 4–8 December 2005. Springer-Verlag.

[51] Ronald L. Rivest. *IETF RFC 1321: The MD5 Message-Digest Algorithm.* Internet Activities Board, April 1992. Available from `http://www.ietf.org/rfc/rfc1321.txt`.

[52] Alexander Russell. Necessary and sufficient conditions for collision-free hashing. *Journal of Cryptology*, 8(2):87–100, 1995.

[53] Ronen Shaltiel. Recent developments in explicit constructions of extractors. *Bulletin of the European Association for Theoretical Computer Science (EATCS)*, 77:67–95, 2002.

[54] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, July and October 1948. Reprinted in D. Slepian, editor, *Key Papers in the Development of Information Theory*, IEEE Press, NY, 1974.

[55] Daniel R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions. In Kaisa Nyberg, editor, *Advances in Cryptology—EUROCRYPT 98*, volume 1403 of *Lecture Notes in Computer Science*. Springer-Verlag, May 31–June 4 1998.

[56] Amnon Ta-Shma and David Zuckerman. Extractor codes. In *ACM Symposium on Theory of Computing*, pages 193–199, 2001.

[57] Luca Trevisan. Notes for Lecture 8. CS294: Pseudorandomness and Combinatorial Constructions. U.C. Berkeley. Available from `http://www.cs.berkeley.edu/~luca/pacc/lecture08.pdf`.

[58] Luca Trevisan. Construction of extractors using pseudo-random generators (extended abstract). In *ACM Symposium on Theory of Computing*, pages 141–148, 1999.

[59] Luca Trevisan, Salil P. Vadhan, and David Zuckerman. Compression of samplable sources. Technical Report TR05-012, Electronic Colloquium on Computational Complexity (ECCC), 2005.

[60] Hoeteck Wee. On pseudoentropy versus compressibility. In *IEEE Conference on Computational Complexity*, pages 29–41. IEEE Computer Society, 2004.

[61] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions. In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91, Chicago, Illinois, 3–5 November 1982. IEEE.

Curriculum Vitae

Chun-Yuan Hsiao

Department of Computer Science
Boston University
111 Cummington Street
Boston, MA 02215

Office: PSY 224
Phone: 617-358-2356
Email: cyhsiao@bu.edu

Research Interests

Cryptography, Security, Theoretical Computer Science. Some specific areas:

- Protocol/Primitive Design
- Computational Entropy, Pseudorandomness
- Hash Functions

Education

- Ph.D. (2002 - 2010), Department of Computer Science, Boston University
  Adviser: Prof. Leonid Reyzin
- M.S. (1999 - 2001), Department of Computer Science and Information Engineering, National Taiwan University
  Advisers: Dr. Chi-Jen Lu. and Prof. Yuh-Dauh Lyuu
- B.S. (1995 - 1999), Department of Computer Science and Information Engineering, National Taiwan University

Employment History

- Fall 2006, Fellow, Institute for Pure and Applied Mathematics (IPAM) at UCLA. (Workshop on Securing Cyberspace: Application and Foundations of Cryptography and Computer Security)
- Fall 2006, 2005, 2004, Spring 2004, Summer 2003, Research Assistant, Department of Computer Science at Boston University. Supervisor: Prof. Leonid Reyzin
- Summer 2006, 2005, 2004, Spring 2002, Research Assistant, Academia Sinica, Taiwan. Supervisor: Dr. Chi-Jen Lu

Publications

- **Conditional Computational Entropy, or Toward Separating Pseudoentropy from Compressibility**, with Chi-Jen Lu and Leonid Reyzin. Advances in Cryptology – *EUROCRYPT 2007*, Moni Naor, editor, Lecture Notes in Computer Science 4515 Springer-Verlag, pages 169-186, 2007.

- **The Impossibility of Basing One-Way Permutations on Central Cryptographic Primitives**, with Yan-Cheng Chang and Chi-Jen Lu. *Journal of Cryptology* 19(1): pages 97-114, 2006.
- **Finding Collisions on a Public Road, or Do Secure Hash Functions Need Secret Coins?** with Leonid Reyzin. Advances in Cryptology – *CRYPTO 2004*, Matt Franklin, editor, Lecture Notes in Computer Science 3152, Springer-Verlag, pages 92-105, 2004.
- **On the Impossibilities of Basing One-Way Permutations on Central Cryptographic Primitives**, with Yan-Cheng Chang and Chi-Jen Lu. Advances in Cryptology – *ASIACRYPT 2002*, Yuliang Zheng, editor, Lecture Notes in Computer Science 2501, Springer-Verlag, pages 110-124, 2002.

Talks and Presentations

- **Conditional Computational Entropy, or Toward Separating Pseudoentropy from Compressibility**
  - June 2007, Cryptography Group Seminar at Brown University
  - May 2007, The 26th Annual Eurocrypt Conference (*EUROCRYPT 2007*)
  - December 2006, Workshop on Securing Cyberspace: Applications and Foundations of Cryptography and Computer Security, held by Institute for Pure and Applied Mathematics at UCLA
- **Finding Collisions on a Public Road, or Do Secure Hash Functions Need Secret Coins?**
  - July 2005, Seminar at Taiwan Information Security Center
  - April 2005, Cryptography and Information Security Group Seminar at MIT
  - August 2004, The 24rd Annual International Cryptology Conference (*CRYPTO 2004*)
- **Private Information Retrieval does not Imply One-Way Permutations**
  - 2003, Applied Cryptography and Electronic Security Group Seminar at Boston University

Honors and Awards

- 2002 - 2006, Boston University Presidential Scholarship
- 1996, National Taiwan University Dean's List
- 1997, 4th place in National Collegiate Programming Contest, Taiwan
- 1996, 12th place in National Collegiate Programming Contest, Taiwan

Teaching Experience

Teaching Fellow at the Department of Computer Science at Boston University

- Spring&Fall 2007, Spring&Fall 2008: CS101, Introduction to Computers (undergraduate)
- Spring 2006, CS530, Advanced Algorithms (graduate)
- Spring 2005, CS330, Analysis of Algorithms (undergraduate)
- Fall 2003, CS535, Complexity (graduate)

References

Prof. Leonid Reyzin
Department of Computer Science
Boston University
111 Cummington Street, Boston, MA 02215, USA
reyzin@cs.bu.edu

Dr. Chi-Jen Lu
Institute of Information Science
Academia Sinica
Nankang 115 Taipei, Taiwan
cjlu@iis.sinica.edu.tw

Dr. Gene Itkis
Lincoln Laboratory
Massachusetts Institute of Technology
244 Wood Street, Lexington, MA 02420-9108, USA
itkis@ll.mit.edu