Thesis for the degree
Doctor of Philosophy

חבור לשם קבלת התואר
דוקטור לפילוסופיה

By
Iftach Haitner

מאת
יפתח הייטנר

New Implications and Improved Efficiency of Constructions Based on One-way Functions

יישומים חדשים ובניות יעילות מבוססות פונקציות חד כיווניות

Advisor
Prof. Omer Reingold

מנחה
פרופ' עמר ריינגולד

March 2008

אדר א' תשס"ח

Submitted to the Scientific Council of the
Weizmann Institute of Science
Rehovot, Israel

מוגש למועצה המדעית של
מכון ויצמן למדע
רחובות, ישראל

To my father, Moti Haitner, who knew little about Science, but much about what it takes to be a Human Being.

Summary

Since most interesting cryptographic tasks are impossible to achieve with absolute, information-theoretic security, modern cryptography aims to design cryptographic primitives (i.e., algorithms/protocols) that are computationally infeasible to break (i.e., secure against computationally bounded adversaries). Proving lower bounds of the type needed, however, seems beyond the reach of current techniques in complexity theory. [1] Thus, research in the *Foundations of Cryptography* has aimed to design primitives based on complexity assumptions that are as weak as possible. It is well known that the assumption that *one-way functions* exist, is necessary for most cryptographic primitives. It is then natural to pose the opposite question. Namely, does the existence of one-way functions imply the existence of all cryptographic primitives? Here we consider some of the most fundamental primitives in cryptography and prove the following results about the power of one-way functions in implementing these primitives.

Statistically hiding commitments. Statistically hiding commitments (ones where the hiding property holds against even computationally unbounded adversaries) are among the few fundamental primitives for which we have failed to find exact characterization. That is, until recently it was only known how to build these primitives from seemingly stronger assumptions than the existence of one-way functions, yet there was no black-box separation between these primitives and one-way functions.

We resolve the complexity of statistically hiding commitments, giving a construction of statistically hiding commitment schemes under the minimal complexity assumption that one-way functions exist. By this we give a positive answer to an open question posed by Naor, Ostrovsky, Venkatesan, and Yung (CRYPTO '92, J. Cryptology '98).

Pseudorandom generators. We present a construction of pseudorandom generators based on regular one-way functions that is significantly better (in terms of security and efficiency) than the previous construction of Goldreich, Krawczyk and Luby (FOCS '88, SIAM J. on Computing '99) (i.e., the input length of our generator is $\Theta(n \log n)$ compared to $\Theta(n^3)$ in Goldreich et al., where n is the input length of the underlying one-way function). In addition, we present a construction of pseudorandom generators from any one-way function that improves the construction of Håstad, Impagliazzo, Levin and Luby (STOC '89, STOC '90, SIAM J. on Computing '99). Finally, we show a rather efficient construction of pseudorandom generator (input length $\Theta(n^2)$) based on one-way functions with exponential hardness. Our construction significantly improves the previous construction due to Holenstein (TCC '06) (input length $\Theta(n^5)$).

[1] Indeed, it would require at least proving that P $\neq$ NP.

Interactive hashing. We give an alternative proof for interactive hashing protocol of Naor, Ostrovsky, Venkatesan and Yung (CRYPTO '92, J. Cryptology '98), which seems to us significantly simpler and more intuitive than the original one. Moreover, the new proof achieves much better parameters (in terms of how security preserving the reduction is). Finally, our proof implies a more versatile interactive hashing theorem in a more general setting than that of Naor et al.

Hardness amplifications. We present a reduction for security amplification of *regular* one-way functions, which incur only $\Theta(n \log(n))$ blow-up in the input length. This improves upon the reduction of Goldreich, Impagliazzo, Levin, Venkatesan and Zuckerman (FOCS '90) in that the reduction does not need to know the regularity parameter of the functions (in terms of security, the two reductions are incomparable).

Acknowledgements

It is a great pleasure to thank the many people who helped me during my thesis.

First and foremost, I wish to thank my advisor Omer Reingold. I am grateful to Omer for his patience and for sharing with me so much of his knowledge and wisdom. During the long conversations we had together, I learned a lot about science, but even more importantly, about how to enjoy science. With Omer's companionship, no task seemed too frightening. While never hesitating to provide sincere and determined opinions, Omer always did it in the most gentle way, teaching me a lot about the right way to deliver your point of view. Finally, I am grateful to Omer for his personal advice and friendship, I will always be in debt for it.

My interest in Cryptography originated during my Masters studies with Oded Goldreich. Cryptography never seems more exciting than while being taught by Oded. Oded guided my first steps as a scientist, and helped me discover the beauty of research. Finally, numerous times through my PhD. studies I came to Oded seeking advice (both on scientific and "non-scientific" issues), and always benefited from his smart and unconventional opinions.

Oded Goldreich and Moni Naor served as my very helpful and joyful interim and final examination committee, and I would like to thank them for that.

Ronen Shaltiel was my first coauthor and was the first to see me as a colleague. Since then, Ronen enriched my education with many of his very intelligent and insightful observations.

Throughout my studies, I have had the fortune of interacting with Yuval Ishai. His deep solid insights, everlasting patience and affection to Cryptography opened up a new world.

During my last year at Weizmann Institute, I have had the fortune of interacting with Shafi Goldwasser. I am very grateful for her patience, and for the long hours she spent helping me to transform my thoughts into more digestible material.

Other researchers that have deeply influenced my research include Cynthia Dwork, Yuval Emek, Danny Harnik, Jonathan Hoch, Thomas Holenstein, Yehuda Lindel, Jonathan Katz, Tal Moran, Moni Naor, Alon Rosen, Gil Segev and Salil Vadhan. I am very grateful for my discussions with them. In particular, I would like thank Danny Harnik for the collaboration that yielded the results presented in Chapters 5 and 6.

I had befitted a lot from the Students Theory Seminar organized by Dana Moshkovitz and from the Crypto Reading Group organized by Shafi Goldwasser, and would like to thank them for their efforts.

I had many happy times during these years at Weizmann Institute. Dan Levi, Shai Litvak and Lior Noy took part in many of them. The "non-scientific" activities in Weizmann Institute had also contributed a lot to my well being, this list includes the chess club, the swimming group, the

Ultimate Frisbee group, and most importantly the Playback Theater group. I in great debt to the people who made these activities possible.

Finally, I would like to thank my wife Liat, for her love and support through these long years. This thesis would never have happened without her.

Contents

Chapter 1

Introduction

1.1 Overview

Since most interesting cryptographic tasks (e.g., key-agreement protocols, encryption schemes) are impossible to achieve with absolute, information-theoretic security, modern cryptography aims to design cryptographic primitives (i.e., algorithms/protocols) that are computationally infeasible to break. Namely, their security is based on computational hardness assumptions. Proving lower bounds of the type needed, however, seems beyond the reach of current techniques in complexity theory, and indeed would require at least proving that P $\neq$ NP. [1]

Given this state of affairs, research in the foundations of cryptography has aimed to design cryptographic protocols based on complexity assumptions that are as weak and general as possible. Typically, complexity assumptions come in two flavors: *specific hardness assumptions* like discrete log, factoring and RSA, and *general hardness assumptions* like the existence of one-way functions and of collision resistant hash functions. Since one-way functions (efficient computable functions that are infeasible to invert) are implied by almost any cryptographic task [IL89, OW93], a large effort was devoted to base the basic cryptographic primitives on their mere existence. This project was enormously successful in the 1980's. In a beautiful sequence of works, it was shown that many cryptographic primitives, such as pseudorandom generators, pseudorandom functions, private-key encryption and authentication, digital signatures, (computationally hiding) commitment schemes, and (computational) zero-knowledge proofs could be constructed from any one-way function [HILL99, GGM86, Rom90, Nao91, GMW91].

Primitives assumed to carry some hardness assumption can be used to construct a provably secure cryptographic tasks in two possible ways: "black-box usage", where the construction uses only the input/output behavior of the primitive (i.e., the primitive is accessed as a "subroutine"), and "non-black-box usage", where the construction uses the internal structure of the primitive, e.g., its code. It was shown that many of the primitives whose existence we failed to reduce to the existence of one-way functions, such as public-key encryption, collision-resistant hashing and oblivious transfer, could not be black-box reduced to the existence of one-way functions [IR89, Sim98].

Few important primitives, however, resisted classification into the above categories. That is, it was only known how to build these primitives from seemingly stronger assumptions than the

[1] For most cryptographic primitives, the task of breaking these primitives can be translated into an NP problem. Thus, proving the security of such primitives against polynomially bounded adversaries, would imply that P $\neq$ NP.

existence of one-way functions, yet there was no black-box separation between these primitives and one-way functions. One of the most fundamental primitive among these few examples is *statistically hiding commitment*. In Section 1.2, we describe this primitive and present our result that resolves the complexity of this primitive, giving a (black-box) construction of statistically hiding commitment schemes under the minimal complexity assumption that one-way functions exist. *Interactive hashing* is a major component in all known constructions of statistically hiding commitment that are based on, possibly restricted types of, one-way functions (in particular, in the one described in this work). We describe this notion in Section 1.3, and present our contribution to efficient constructions of interactive hashing protocols.

A closely related line of research to the one discussed above, tries to find the most *efficient* construction of a cryptographic primitive based on a given complexity assumption. In particular, what is the most efficient construction assuming the existence of (particular type of) one-way functions. While in some settings very efficient constructions where given (i.e., hardness amplification of one-way permutations [GIL+90]), others constructions are highly inefficient, making them impractical in practice. Here again, lower bounds on the efficiency of any black-box reduction were given [KST99, GGKT05, HHRS07, HK05, Wee07]. The above question is of high importance w.r.t. the fundamental notions of *pseudorandom generators* and of *hardness amplification*. In Section 1.4, we describe pseudorandom generators and present our efficient constructions of these primitives from different types of one-way functions. Where in Section 1.5, we discuss the notion of hardness amplification and present our efficient constructions.

1.2 Statistically Hiding Commitment

A commitment scheme defines a two-stage interactive protocol between a sender S and a receiver R; informally, after the *commit stage*, S is bound to (at most) one value, which stays hidden from R, and in the *reveal stage* R learns this value. The two security properties hinted at in this informal description are known as *binding* (namely, that S is bound to at most one value after the commit stage) and *hiding* (namely, that R does not learn the value to which S commits before the reveal stage).

As with most cryptographic primitives, each of these security properties comes in two main flavors — *computational security,* whereby a polynomial-time adversary cannot violate the property except with negligible probability, and the stronger notion of *statistical security*, whereby even a computationally unbounded adversary cannot violate the property except with negligible probability. (An even stronger notion is that of *perfect security*, in which we do not even allow a negligible probability of breaking the scheme.) Naturally, statistical security, when achievable, is preferable to computational security. However, it can be shown that there do not exist commitment schemes that are simultaneously statistically hiding and statistically binding. Thus, at best we can hope for one of the two properties to be statistical and the other to be computational. The complexity of *statistically binding* commitment schemes has been understood for a long time; they can be constructed from any one-way function [Nao91, HILL99] and conversely, one-way functions are necessary for commitment schemes, even with both security properties computational [IL89]. In this work, however, we are interested in *statistically hiding* commitments.

Some of the most important examples of cryptographic protocols based on commitments are the zero-knowledge protocols for proving membership in an arbitrary NP language [GMW91, BCC88]. In the protocol of [GMW91], the hiding property of the commitment scheme translates to the

zero-knowledge property of the protocol (i.e. the verifier learns nothing other than the fact that the assertion being proven is true), and the binding property of the commitment translates to the soundness property of the protocol, (i.e. the prover cannot convince the verifier of a false assertion). Thus, the existence of statistically hiding commitments implies that arbitrary NP statements can be proven with statistical zero knowledge and computational soundness; that is, every language in NP has a *statistical zero-knowledge argument system* [BCC88, BCY91, NOVY98].

Using statistically hiding commitments and the resulting statistical zero-knowledge arguments in known reductions [GMW91, GMW87], one can actually transform *any* two-party protocol that provides statistical security for one of the parties against a passive (a.k.a. honest-but-curious) adversary into one that provides statistical security for the same party against a malicious adversary (while preserving computational security for the other party).

1.2.1 Our Results

We resolve the complexity of statistically hiding commitments, giving a (black-box) construction of statistically hiding commitment schemes under the minimal complexity assumption that one-way functions exist. By this we give a positive answer to an open question posed by Naor, Ostrovsky, Venkatesan, and Yung (CRYPTO '92, J. Cryptology '98).

1.3 Interactive Hashing

Interactive hashing, introduced by Naor, Ostrovsky, Venkatesan and Yung [NOVY98], is a protocol that allows a sender S to commit to a particular value while only reviling to a receiver R some predefined information of this value. More specifically, S commits to a value y while only reviling to R the value $(h, h(y))$, where h is some random hash function. The two security properties of interactive hashing are *binding* (namely, S is bounded by the protocol to at most one value of y) and *hiding* (namely, R does not learn any impermissible information about y). As in [NOVY98], we will consider in this work interactive hashing where the hiding property is statistical (i.e., the protocol preserves the secrecy of y even against an all-powerful R), and the binding property is computational (i.e., it assumes that S is computationally bounded).

Interactive hashing (in the flavor mentioned above) is closely related and to a large extent motivated by the notion of statistically hiding commitments. The relation between interactive hashing and statistically hiding commitments goes beyond the similarity in definitions. On one hand, interactive hashing can easily be implemented using commitment schemes (simply commit to y using the commitment scheme and reveal whatever information needed on y in the clear). On the other hand, one of the main applications of interactive hashing protocols is for constructing statistically hiding commitment schemes. A major motivation for looking into interactive hashing, is to simplify constructions of statistically hiding commitment schemes based on any one-way functions mentioned above.

1.3.1 Our Results

We introduce an alternative proof for the [NOVY98] protocol, which relies in parts on the original proof due to [NOVY98], but still seems to us significantly simpler. Moreover, the parameters achieved by our proof are an improvement compared with the original ones. Given an algorithm A that breaks the binding property with probability ε_A, we get an algorithm that inverts the one-way

permutation in comparable time and with inverting probability $\varepsilon_A^2 \cdot \text{poly}(n)$. This is a substantial improvement and seems much closer to natural limitations of the proof technique. [2]

In addition to being simpler and more security preserving, the new proof implies a more general interactive hashing theorem. The new theorem applies to every family of hash functions that is a product of Boolean families of pairwise independent hash functions (and not only to the special family of two-to-one hash functions used by [NOVY98, NOV06]). More importantly, the new theorem directly applies to the "sparse case", where the set we apply the hash function on might be negligible in the set of all strings.

1.4 Pseudorandom Generators

Pseudorandom Generators, a notion first introduced by Blum and Micali [BM82] and stated in its current, equivalent form by Yao [Yao82], are one of the cornerstones of cryptography. Informally, a pseudorandom generator is a polynomial-time computable function G that stretches a short random string x into a long string $G(x)$ that "looks" random to any efficient (i.e., polynomial-time) algorithm. Hence, there is no efficient algorithm that can distinguish between $G(x)$ and a truly random string of length $|G(x)|$ with more than a negligible probability. Originally introduced in order to convert a small amount of randomness into a much larger number of effectively random bits, pseudorandom generators have since proved to be valuable components for various cryptographic applications, such as bit commitments [Nao91], pseudorandom functions [GGM86] and pseudorandom permutations [LR88], to name a few.

The first construction of a pseudorandom generator was given in [BM82] based on a particular one-way function and was later generalized in [Yao82] into a construction of a pseudorandom generator based on any one-way permutation. Their result was generalized to by Goldreich et al. [GKL93] that showed a construction of a pseudorandom generator based on any regular one-way function. (A regular function is a function such that every element in its image has the same number of preimages). Finally, Håstad et al. [HILL99] (combining [ILL89a, Hås90]), culminated this line of research by showing a construction of a pseudorandom generator using any one-way function. Recently, Holenstein [Hol06a] generalized the construction of [HILL99], to get more efficient pseudorandom generators from any one-way function with *exponential hardness* (i.e., for some constant C, no algorithm of running-time at most 2^{Cn} inverts the function with probability better than 2^{-Cn}).

The Complexity and Security of the Previous Constructions

While the HILL generator fully answers the question of the plausibility of a generator based on any one-way function, the construction is highly involved and very inefficient. Other than the evident contrast between the simplicity and elegance of the BMY generator to the complex construction and proof of the HILL generator, the parameters achieved in the construction are far worse, rendering the construction impractical.

In practice, it is not necessarily sufficient that a reduction translates polynomial security into polynomial security. In order for reductions to be of any practical use, the concrete overhead introduced by the reduction comes into play. There are various factors involved in determining the security of a reduction, and in Section 2.3.2 we elaborate on the security of cryptographic reductions

[2] We note that independently of our work, [NOV06] recently presented an $\varepsilon_A^3 \cdot \text{poly}(n)$ reduction.

and the classification of reductions in terms of their security. Here, however, we focus only on one central parameter, which is the length m of the generator's seed compared to the length n of the input to the underlying one-way function. The BMY generator takes a seed of length $m = \Theta(n)$, the GKL generator takes a seed of length $m = \Theta(n^3)$ while the HILL construction produces a generator with seed length on the order of $m = \Theta(n^8)$.[3]

The length of the seed is of great importance to the security of the resulting generator. While it is not the only parameter, it serves as a lower bound to how good the security may be. For instance, the HILL generator on m bits has security that is at best comparable to the security of the underlying one-way function, but only on $\Theta(\sqrt[8]{m})$ bits. To illustrate the implications of this deterioration in security, consider the following example: Suppose that we only trust a one-way function when applied to inputs of at least 100 bits, then the GKL generator can only be trusted when applied to a seed of length of at least one million bits, while the HILL generator can only be trusted on seed lengths of 10^{16} and up (both being highly impractical). Thus, trying to improve the seed length towards a linear one (as it is in the BMY generator) is of great importance in making these constructions practical.

1.4.1 Our Results

We give a construction of a pseudorandom generator from any regular one-way function with seed length $O(n \log n)$. We note that our approach has the potential of reaching a construction with a linear seed, the bottleneck being the efficiency of the current bounded-space generators. Our construction is achieved in two steps:

- We give a significantly simpler proof that the generator of [GKL93] works, allowing the use of a family of hash functions which is pairwise-independent rather than n-wise independent (as used in [GKL93]). This gives a construction with seed length $O(n^2)$.
- The new proof allows for the derandomization of the choice of the randomizing hash functions via the *bounded-space generator* of Nisan [Nis92], further reducing the seed length to $O(n \log n)$.

We give a construction of a pseudorandom generator from any exponentially hard one-way function with seed length $O(n^2)$. If the resulting generator is allowed to have only super-polynomial security then the construction gives seed length of only $O(n \log^2 n)$. Finally, we present a pseudorandom generator from *any* one-way function with seed length $O(n^7)$, which is the best known to date. Our improvements to the seed length of pseudorandom generators under the various assumptions are summarized in Figure 1.1.

1.5 Hardness Amplification

The existence of one-way functions is essential to almost any task in cryptography (see for example [IL89]) and also sufficient for numerous cryptographic primitives, such as the pseudorandom generators discussed above. In general, for constructions based on one-way functions we use what are

[3] The seed length actually proved in [HILL99] is $\Theta(n^{10})$, however it is mentioned that a more careful analysis can get to $\Theta(n^8)$. A formal proof for the $\Theta(n^8)$ seed length construction is given by Holenstein [Hol06a].

Paper	Type of function	Seed length
[BM82, Yao82]	One-way permutation	$\Theta(n)$
[GKL93]	Regular one-way function	$\Theta(n^3)$
This work		$\Theta(n \log n)$
[Hol06a]	One-way function with exponential hardness	$\Theta(n^5)$
This work		$\Theta(n^2)$
This work	Regular one-way function with exponential hardness	$\Theta(n)$
[HILL99]	Any one-way function	$\Theta(n^8)$
This work		$\Theta(n^7)$

Figure 1.1: Our improvements to pseudorandom generators.

called *strong* one-way functions. That is, functions that can only be inverted efficiently with negligible success probability. A more relaxed definition is that of an δ-weak one-way function, where $\delta(n)$ is a polynomial fraction. This is a function that no efficient algorithm inverts with probability better than $1-\delta(n)$. This definition is significantly weaker, yet Yao [Yao82] showed how to convert any weak one-way function into a strong one. The new strong one-way function simply consists of many independent copies of the weak function concatenated to each other. The solution of Yao, however, incurs a blow-up factor of at least $\omega(1)/\delta(n)$ to the input length of the strong function, [4] which translates to a significant loss in the security (as in the case of pseudorandom generators).

With this security loss in mind, several works have tried to present an efficient method of amplification from weak to strong. Goldreich et al. [GIL+90] give a solution for one-way permutations that has just a linear blowup in the length of the input. This solution generalizes to *known-regular* one-way functions (regular functions whose image size is efficiently computable), where its input length varies according to the required security. The input length is linear when security is at most $2^{\Omega(\sqrt{n})}$, but deteriorates up to $O(n^2)$ when the required security is higher (e.g., security $2^{O(n)}$). [5]

1.5.1 Our Results

We present an alternative efficient hardness amplification for regular one-way functions. Our construction is arguably simpler and has the following advantages:

1. While the [GIL+90] construction works only for *known* regular weak one-way functions, our amplification works for any regular weak one-way functions (whether its image size is efficiently computable or not).

2. The input length of the resulting strong one-way function is $O(n \log n)$ regardless of the required security. Thus, for some range of the parameters our solution is better than that of [GIL+90] (although it is worse than [GIL+90] for other ranges).

[4] The $\omega(1)$ factor stands for the logarithm of the required security. For example, if the security is $2^{O(n)}$ then this factor is of order n.

[5] Loosely speaking, one can think of the security as the probability of finding an inverse to a random image $f(x)$ simply by choosing a random element in the domain.

1.6 Outline

We present the basic notations and definitions in Chapter 2, additional definitions are given in the chapters themselves. Chapter 2 also contains some of the common tools used through this work, some of these tools are original contributions of this thesis. In Chapter 3 we give our construction of statistically hiding commitments from any one-way function, where in Chapter 4 we give our new interactive hashing theorem. In Chapter 5 we present our contribution to efficient constructions of pseudorandom generators. Finally in Chapter 6 we use some of the tools developed in Chapter 5 to obtain our efficient hardness amplification result.

Chapter 2

Preliminaries

2.1 General Notations

Given two strings x and y, we denote their concatenation by $x \circ y$ and denote their inner product modulus two by $\langle x, y \rangle_2$. Given a function $f : \{0,1\}^* \mapsto \{0,1\}^*$ and a set $L \subseteq \{0,1\}^*$, we denote the image of f on L as $f(L) \stackrel{\text{def}}{=} \{f(x) : x \in L\}$ and denote $f(\{0,1\}^*)$ by $Im(f)$. For $y \in Im(f)$, we define $f^{-1}(y) \stackrel{\text{def}}{=} \{x \in \{0,1\}^* : f(x) = y\}$. The ***degeneracy*** of f on y is defined by $\mathrm{Deg}_f(y) \stackrel{\text{def}}{=} \lceil \log |f^{-1}(y)| \rceil$. We denote by $f : \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}$, the ensemble of functions $\{f_n : \{0,1\}^n \to \{0,1\}^{\ell(n)}\}_{n \in \mathbb{N}}$. A function $\mu : \mathbb{N} \to [0,1]$ is called *negligible* if $\mu(n) = n^{-\omega(1)}$. We let $\mathrm{neg}(n)$ denote an arbitrary negligible function (i.e., when we say that $f(n) < \mathrm{neg}(n)$ we mean that *there exists* a negligible function $\mu(n)$ such that for every n, $f(n) < \mu(n)$). Likewise, $\mathrm{poly}(n)$ denotes an arbitrary function $f(n) = n^{O(1)}$, where PPT refers to probabilistic algorithms (i.e., Turing machines) that run in *strict* polynomial time.

2.1.1 Interactive Protocols

An *interactive protocol* (A, B) consists of two algorithms that compute the next-message functions of the (honest) parties in the protocol. Specifically, $A(x, a, \alpha_1, \ldots, \alpha_k; r)$ denotes the next message α_{k+1} sent by party A when the common input is x, A's auxiliary input is a, A's coin tosses are r, and the messages exchanged so far are $\alpha_1, \ldots, \alpha_k$. There is a special messages, `halt`, that immediately halts the interaction, at which time each party can compute one more message, which is their *private output*.

We write $(A(a), B(b))(x)$ to denote the random process obtained by having A and B interact on common input x, with (private) auxiliary inputs a and b to A and B, respectively (if any), and with independent random coin tosses for A and B. We call (A, B) *polynomially bounded* if there is a polynomial p such that for all x, a, b, the running-time each party is at most $p(|x|)$ with probability one. Moreover, if B^* is any interactive algorithm, then A will immediately halt in $(A(a), B^*(b))(x)$ if its running-time exceeds $p(|x|)$; we have the analogous requirement for B interacting with any A^*. The number of *rounds* in an execution of the protocol is the *total* number of messages exchanged between A and B. We call the protocol (A, B) *public coin* for A (resp., B) if all the messages sent by A (resp., B) are simply the output of its coin tosses (independent of the history), except for the final `halt` message and A's (resp., B's) private output, which is computed as a deterministic function of the transcript.

We associate several random variables with the interaction $(A(a), B(b))(x)$. The private output of A is denoted by $\text{output}_A(A(a), B(b))(x)$, and $\text{view}_A(A(a), B(b))(x)$ denotes A's *view* of the interaction, i.e., its values are transcripts $(\gamma_1, \gamma_2, \ldots, \gamma_t; r)$, where the γ_i's are all the messages exchanged and r is A's coin tosses. Similarly, $\text{output}_B(A(a), B(b))(x)$ and $\text{view}_B(A(a), B(b))$ denote B's private output and view, respectively. The joint output, if any, is denoted by $\text{output}(A(a), B(b))(x)$.

2.1.2 Distributions and Entropy

Given a random variable X taking values in a finite set $\mathcal{U}$, we write $x \leftarrow X$ to indicate that x is selected according to X. Given a subset S of $\mathcal{U}$, we let $x \leftarrow S$ denote that x is selected according to the uniform distribution on S. We adopt the convention that when the same random variable occurs several times in an expression, all occurrences refer to a single sample. For example, $\Pr[f(X) = X]$ is defined to be the probability that when $x \leftarrow X$, we have $f(x) = x$. We write U_n to denote the random variable distributed uniformly over $\{0,1\}^n$. Let D be a distribution over the set L, the support of D is defined as: $\text{Supp}(D) \stackrel{\text{def}}{=} \{x \in L : D(x) > 0\}$. We write X^k to denote the random variable consisting of k independent copies of X. For an event E, $X|_E$ denotes the random variable X conditioned on E. The *statistical difference* (also known as the *variation distance*) between random variables X and Y taking values in $\mathcal{U}$ is defined to be $\Delta(X,Y) = \max_{S \subset \mathcal{U}} |\Pr[X \in S] - \Pr[Y \in S]|$. We define the distinguishing advantage of an algorithm A, on security parameter n, between X and Y by

$$\Delta_n^A(X,Y) = |\Pr[A(1^n, X) = 1] - \Pr[A(1^n, Y_n) = 1]|,$$

where the probabilities are taken over the distributions X and Y, and the randomness of A (we omit the security parameter from the above notation whenever its value is clear from the context). By a *Distribution Ensemble* we mean a series $\{D_n\}_{n \in \mathbb{N}}$, where each D_n is a distribution over a finite set $\mathcal{U}_n$. Two probability ensembles $\{X_n\}_{n \in \mathbb{N}}$ and $\{Y_n\}_{n \in \mathbb{N}}$ are *computationally indistinguishable* if $\Delta^A(X_n, Y_n) < \text{neg}(n)$ for every PPT A. Similarly, we say that $\{X_n\}$ and $\{Y_n\}$ are *statistically indistinguishable* if the above is required for all functions A (instead of only PPT A's), or equivalently if $\Delta(X_n, Y_n) < \text{neg}(n)$. Let D be a distribution over some finite domain X, we use the following measures of entropy:

- The Shannon-entropy of D is $\text{H}(D) = \sum_{x \in X} D(x) \log \frac{1}{D(x)}$.
- The collision-probability of D is $\text{CP}(D) = \sum_{x \in X} D(x)^2$.
- The min-entropy of D is $\text{H}_\infty(D) = min_{x \in X} \log \frac{1}{D(x)}$.

2.2 Pairwise Independent Hash Functions

DEFINITION 2.2.1 (pairwise Independent hash functions)
Let $\mathcal{H}$ be a family of functions mapping strings of length n to strings of length $\ell(n)$. We say that $\mathcal{H}$ is an efficient family of pairwise independent hash functions (following [CW79]) if the following hold: [1]

[1] The first two properties, regarding the efficiency of the family, implicitly assume an ensemble of families (one family for every value of n). For simplify of presentation, we only refer to a single family.

Samplable. $\mathcal{H}$ is polynomially samplable in n,

Efficient. There exists a polynomial-time algorithm that given $x \in \{0,1\}^n$ and a description of $h \in \mathcal{H}$ outputs $h(x)$,

Pairwise independence. For every distinct $x_1, x_2 \in \{0,1\}^n$ and every $y_1, y_2 \in \{0,1\}^{\ell(n)}$, we have

$$\Pr_{h \leftarrow \mathcal{H}}[h(x_1) = y_1 \wedge h(x_2) = y_2] = 2^{-2\ell(n)} \ .$$

It is well known ([CW79]) that there exists an efficient family of pairwise-independent hash functions for every $\ell(n) \in \text{poly}(n)$, whose elements description size is $O(\max\{n, \ell(n)\})$.

Given a family of hash functions, we sometime consider the concatenation of this family to itself defined next.

DEFINITION 2.2.2 (product hash family)
Let $\mathcal{H}$ be a family of functions mapping strings of length n to strings of length $\ell(n)$ and let $k : \mathbb{N} \mapsto \mathbb{N}$. The k-product-family of $\mathcal{H}$, denoted $\mathcal{H}^{k(n)}$, is a family of functions mapping strings of length n to strings of length $k(n)\ell(n)$ which is defined as follows: The members of $\mathcal{H}^{k(n)}$ are all possible tuples $\overline{h}$ of $k(n)$ functions from $\mathcal{H}$. For every such tuple $\overline{h} = (\overline{h}_1, \ldots, \overline{h}_{k(n)})$ and every $x \in \{0,1\}^n$ we let $\overline{h}(x) = (\overline{h}_1(x), \ldots, \overline{h}_{k(n)}(x))$.

Through this work we make use of the following facts about pairwise independent hash functions. The following standard lemma (see for example, [Gol01b, Lemma 4.3.1]) states that a random pairwise independent hash function partitions a given set into (almost) equal size subsets.

LEMMA 2.2.3
Let $\mathcal{H}$ be a family of pairwise independent hash functions mapping strings of length n to strings of length ℓ, let $L \subseteq \{0,1\}^n$ and let $\mu = |L|/2^\ell$. Then for every $y \in \{0,1\}^\ell$ and $\delta > 0$, it holds that

$$\Pr_{h \leftarrow \mathcal{H}}[||\{x \in L \ : \ h(x) = y\}| - \mu| > \delta\mu] < \tfrac{1}{\delta^2 \mu} \ .$$

Finally, we also make use of the known Leftover Hash Lemma.

LEMMA 2.2.4 (Leftover Hash Lemma [BBR88, ILL89b])
Let random variable H denote a uniformly random hash function from a family of pairwise-independent hash functions $\mathcal{H}$ mapping n-bit strings to ℓ-bit strings, and let X be a random variable taking values in $\{0,1\}^n$. For any $\varepsilon > 0$, if $\mathrm{H}_\infty(X) \geq \ell + 2\log(1/\varepsilon)$, and H is independent from X, then the random variable $(H, H(X))$ is ε-close in statistical distance to uniform. [2]

2.2.1 Almost Pairwise Independent Hash Functions

In some cases we cannot afford to use hash functions whose description length is linear in the input size but can afford a description that is linear in the output size. In such cases we use the following

[2] That is, $\text{Ext} : \mathcal{H} \times \{0,1\}^n \mapsto \{0,1\}^\ell$ defined as $\text{Ext}(h, x) = h(x)$, is an explicit $(\ell + 2\log(1/\varepsilon), \varepsilon)$-strong extractor (see Section 2.3 for the definition of extractors).

relaxation of pairwise-independent hash functions.

DEFINITION 2.2.5 (almost pairwise independent hash functions)
Let $\mathcal{H}$ be a family of functions mapping strings of length n to strings of length $\ell(n)$ and let $\delta : \mathbb{N} \mapsto [0,1]$. We say that $\mathcal{H}$ is an efficient family of δ-almost pairwise independent hash functions (following [CW79]) if the following hold:

Samplable. $\mathcal{H}$ is polynomially samplable in n,

Efficient. There exists a polynomial-time algorithm that given $x \in \{0,1\}^n$ and a description of $h \in \mathcal{H}$ outputs $h(x)$,

Almost pairwise independence. For every distinct $x_1, x_2 \in \{0,1\}^n$ and ever $y_1, y_2 \in \{0,1\}^{\ell(n)}$, we have

$$\left|\Pr_{h \leftarrow \mathcal{H}}[h(x_1) = y_1 \wedge h(x_2) = y_2] - 2^{-2\ell(n)}\right| \leq \delta(n) \ .$$

Due to [CW79], [WC81] and [NN93] there exist constructions of efficient families almost pairwise-independent hash functions for every $\ell(n) \in \text{poly}(n)$, whose description length is $O(\log(n) + \ell(n) + \log(1/\delta(n)))$.

2.3 Randomness Extractors

Randomness extractors, introduced by Nisan and Zuckerman [NZ96] are an information theoretic tool for obtaining true randomness from a "weak" source of randomness. In this work, extractors are used in a computational setting to extract pseudorandomness from an imperfect source.

DEFINITION 2.3.1 (strong extractors)
A polynomial-time computable function $\text{Ext} : \{0,1\}^d \times \{0,1\}^n \mapsto \{0,1\}^\ell$ is an (explicit) (k,ε)-strong extractor if for every distribution X over $\{0,1\}^n$ with $\text{H}_\infty(X) \geq k$, the distribution $(\text{Ext}(U_d, X), U_d)$ is ε-close to (U_ℓ, U_d).

2.3.1 One-way Functions

DEFINITION 2.3.2 (one-way functions)
Let $t : \mathbb{N} \mapsto \mathbb{N}$ and $\delta : \mathbb{N} \mapsto [0,1]$. A polynomial-time computable function $f : \{0,1\}^* \mapsto \{0,1\}^*$ is $(t(n), \delta(n))$-one-way, if for every algorithm A of running-time at most $t(n)$

$$\Pr[A(1^n, f(U_n)) \in f^{-1}(f(U_n))] < \delta(n) \ .$$

In the case that $\delta(n) = \frac{1}{t(n)}$, we simply write that f is a $t(n)$-one-way. f is one-way if it is $p(n)$-one-way for every polynomial p, and exponentially hard if it is 2^{Cn}-one-way for some constant $C > 0$. A one-way permutation is a one-way function that is a permutation over any input length n. Finally, if f is $(t(n), 1 - \delta(n))$-one-way for $\delta > 1/\text{poly}(n)$ and every polynomial t, it is customary to call f a δ-weak one-way function.

DEFINITION 2.3.3 (regular one-way functions)
Let $f : \{0,1\}^* \mapsto \{0,1\}^*$ be a $(t(n), \delta(n))$-one-way function. f is regular if there exist a function $\alpha : \mathbb{N} \mapsto \mathbb{N}$ such that for every $n \in \mathbb{N}$ and every $x \in \{0,1\}^n$ we have

$$\left|f^{-1}(f(x))\right| = \alpha(n)$$

In the special case that α is also polynomial-time computable, f is known-regular. In this work we do not require this property and our results hold for functions with unknown-regularity. Thus, when we say regular functions we actually mean unknown-regular functions.

Few convention remarks. When the value of the security-parameter (i.e., 1^n) is clear, we allow ourselves to omit it from the adversary's parameters list. Since any one-way function is without loss of generality length-regular (i.e., inputs of same length are mapped to outputs of the same length), it can be viewed as an ensemble of functions mapping inputs of a given length to outputs of some polynomial (in the input) length. Therefore, we can write let $f : \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}$ be a one-way function, where $\ell(n)$ is some polynomial-computable function.

Length Preserving One-way Functions

In the following we prove the "folklore" fact that when given a one-way function it can be assumed, without loss of generality, that it is a length-preserving one. In the case that $\ell(n) < n$ one can generate a length preserving one-way function simply by padding the output with extra zeros. In the case that $\ell(n) > n$, however, one needs to be more careful in order for the input length to remain of the same order.

LEMMA 2.3.4
Let $f : \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}$ be a $(t(n), \delta(n))$-one-way function and let $\mathcal{H}$ be an efficient family of 2^{-2n}-almost pairwise-independent hash functions from $\{0,1\}^{\ell(n)}$ to $\{0,1\}^{2n}$. Define g as follows:

$$g(x_a, x_b, h) = (h(f(x_a)), h)$$

where $x_a, x_b \in \{0,1\}^n$ and $h \in \mathcal{H}$. There exists a polynomial p such that g is a length-preserving $(t(n) - p(n), \delta(n) + 2^{-n+1})$-one-way function.[3]

Proof. The function g is length preserving as both input and output are of length $2n$ plus the description of $h \in \mathcal{H}$. Let A be an algorithm that runs in time $t_A(n)$ and inverts g with probability $\varepsilon_A(n)$. Note that x_b is a dummy input (used just for padding) so the success of A is taken over (x_a, h) and A's randomness. Define B^A as follows:

ALGORITHM 2.3.5
Algorithm B^A for inverting f.

[3] Since a member of $\mathcal{H}$ can be describes using $\Theta(n)$ bits, the input length of g (and therefore also its output length) is in the same order of that of f.

Input: $y \in \mathrm{Im}(f)$.

1. Choose a uniformly random $h \in \mathcal{H}$.
2. Apply $A(h(y), h)$ to get an output (x_a, x_b, h).
3. Output x_a.

Let $p(n)$ be the an upper bound on the sampling and evaluation time of h. Clearly the running-time of B^A is at most $t_A(n) + p(n)$. Algorithm B^A is sure to succeed on any choice of (y, h) so that A succeeds on $(h(y), h)$ and in addition there exists no $z \in \mathrm{Im}(f)$ such that $h(z) = h(y)$ (h does not introduce any collision to y). Let $Col = \{(y, h) \in \{0,1\}^n \times \mathcal{H} \ : \ \exists z \in \mathrm{Im}(f) \ : \ h(z) = h(y)\}$. Thus,

$$\begin{aligned} \Pr[B^A(f(U_n)) \in f^{-1}(f(U_n))] &\geq \Pr[A(g(U_n, H)) \in g^{-1}(g(U_n, H)) \bigwedge g(U_n, H) \notin Col] \\ &\geq \Pr[A(g(U_n, H)) \in g^{-1}(g(U_n, H))] - \Pr[g(U_n, H) \in Col] \ , \end{aligned}$$

where H is a random variable uniformly distributed over $\mathcal{H}$. For all $y, z \in \mathrm{Im}(f)$ the almost pairwise-independence of h yields that the probability that $H(z) = H(y)$ is at most 2^{-2n+1}. Since the size of $\mathrm{Im}(f)$ in $\{0,1\}^{\ell(n)}$ is at most 2^{n+1},a union bound over all possible $z \in \mathrm{Im}(f)$ gives that for any $y \in \mathrm{Im}(f)$ it holds that $\Pr[\exists z \in \mathrm{Im}(f) \ : \ H(z) = H(y)] \leq 2^{-n+1}$. Therefore, by an averaging argument $\Pr[(f(U_n), H) \in Col] \leq 2^{-n+1}$. Putting it all together we get that $\Pr[B^A(f(U_n)) \in f^{-1}(f(U_n))] \geq \varepsilon_A(n) - 2^{-n+1}$. □

2.3.2 The Security of Cryptographic Constructions

Typically the proof of security for cryptographic constructions is based on reductions. In this paradigm we use a presumably secure implementation of one primitive (or possibly several primitives) in order to implement a second primitive. The proof of security for the second primitive relies on the security assumption for the original one. More precisely, we prove that any efficient adversary that breaks the implementation of the second primitive can be used to efficiently break the original primitive. Note that the meaning of "breaking a primitive" and, furthermore, the definition of the *success probability* of an adversary in breaking the primitive, varies between different primitives. For example, in the case of one-way functions the success probability is the fraction of inputs on which the adversary manages to invert the function. Usually, there is a tradeoff between the running-time of an adversary and its success probability (e.g., it may be possible to utterly break a primitive by enumerating all possibilities for the secret key). Therefore, both the running-time and success probability of possible adversaries are relevant when analyzing the security of a primitive. A useful, combined parameter is the *time-success ratio* of an adversary which we define next.

DEFINITION 2.3.6 (time-success ratio)
Let P be a primitive and let A be an adversary running in time $t_A(n)$ and breaking P with probability $\varepsilon_A(n)$. The time-success ratio of A in breaking P is defined as $R(n) = \frac{t_A(n)}{\varepsilon_A(n)}$, where n is the security-parameter of the primitive.[4]

[4] It is convenient to define the security-parameter of a primitive as its input length. This is in particular the convention for the primitives discussed in this work.

Note that in the above definition, the smaller the R the better A is in breaking P. A quantitative analysis of the security of a reduction is crucial for both theoretical and practical reasons. Given an implementation of primitive P using primitive Q along with a proof of security, let R_P be the security-ratio of a given adversary w.r.t. P and let R_Q be the security-ratio of the adversary that the proof of security yields. A natural way to measure the security of a reduction is by the relation between R_P and R_Q. Clearly, the smaller the R_Q comparing to R_P, the better the performance of the adversary the reduction yields when trying to break Q comparing to the performance of the adversary trying to break P.

The most desirable reductions is when $R_Q(n) \in n^{O(1)}O(R_P(O(n)))$. In such reductions, known as *linear-preserving reductions*, we are guaranteed that breaking the constructed primitive is essentially as hard as breaking the original one. Next we find the *polynomial-preserving reductions* when $R_Q \in n^{O(1)}O(R_P(O(n))^{O(1)})$. Note that a linear/polynomial-preserving reduction typically means that the inputs of Q and P are of the same length (up to a constant-ratio). The other side of the scale is when $R_Q \in n^{O(1)}O(R_P(n^{O(1)}))$. In such reductions, known as *weak-preserving reductions*, we are only guaranteed that breaking P is as hard as breaking Q for polynomially smaller security-parameter (e.g., polynomially smaller input length). For a more comprehensive discussion of the above issues the reader may refer to [Gol00, HL92].

Chapter 3

Statistically Hiding Commitments From Any One-Way Function

3.1 Introduction

A commitment scheme defines a two-stage interactive protocol between a sender S and a receiver R; informally, after the *commit stage*, S is bound to (at most) one value, which stays hidden from R, and in the *reveal stage* R learns this value. The two security properties hinted at in this informal description are known as *binding* (namely, that S is bound to at most one value after the commit stage) and *hiding* (namely, that R does not learn the value to which S commits before the reveal stage).

As with most cryptographic primitives, each of these security properties comes in two main flavors — *computational security,* whereby a polynomial-time adversary cannot violate the property except with negligible probability, and the stronger notion of *statistical security*, whereby even a computationally unbounded adversary cannot violate the property except with negligible probability. (An even stronger notion is that of *perfect security*, in which we do not even allow a negligible probability of breaking the scheme.) Naturally, statistical security, when achievable, is preferable to computational security. However, it can be shown that there do not exist commitment schemes that are simultaneously statistically hiding and statistically binding. Thus, at best we can hope for one of the two properties to be statistical and the other to be computational.

The complexity of *statistically binding* commitment schemes has been understood for a long time; they can be constructed from any one-way function [Nao91, HILL99] and conversely, one-way functions are necessary for commitment schemes, even with both security properties computational [IL89]. In this work, however, we are interested in *statistically hiding* commitments, which have some advantages over statistically binding commitments. Specifically, when commitment schemes are used in constructing larger protocols, one typically needs the binding property to ensure the integrity of commitments that are opened *during* the protocol execution itself, and the hiding property to ensure that the unopened commitments remain secret even *after* the protocol execution. Thus, for the binding property, we need only be concerned with the adversary's current resources, and thus it may be safe for this property to be computational. For the hiding property, however, we need to consider resources that the adversary may gain far into the future, and thus statistical security is preferable.

Some of the most important examples of cryptographic protocols based on commitments are the

zero-knowledge protocols for proving membership in an arbitrary NP language [GMW91, BCC88]. In the protocol of [GMW91], the hiding property of the commitment scheme translates to the zero-knowledge property of the protocol (i.e. the verifier learns nothing other than the fact that the assertion being proven is true), and the binding property of the commitment translates to the soundness property of the protocol, (i.e. the prover cannot convince the verifier of a false assertion). Thus, the existence of statistically hiding commitments implies that arbitrary NP statements can be proven with statistical zero knowledge and computational soundness; that is, every language in NP has a *statistical zero-knowledge argument system* [BCC88, BCY91, NOVY98].

Using statistically hiding commitments and the resulting statistical zero-knowledge arguments in known reductions [GMW91, GMW87], one can actually transform *any* two-party protocol that provides statistical security for one of the parties against a passive (a.k.a. honest-but-curious) adversary into one that provides statistical security for the same party against a malicious adversary (while preserving computational security for the other party).

Perfectly hiding commitment schemes and perfect zero-knowledge arguments for NP were first shown to exist based on specific number-theoretic assumptions [BCC88, BKK90, BCY91, CDG87, Ped91] or, more generally, based on any collection of claw-free permutations [GMR88, GK96]. The assumption for statistically hiding commitment schemes and statistical zero-knowledge arguments was reduced further to collision-resistant hash functions [NY89, DPP98]. Even though it seems intuitive that the computational binding property of statistically hiding commitments should be closely related to collision resistance, the beautiful work of Naor, Ostrovsky, Venkatesan, and Yung [NOVY98] showed that actually any one-way permutation can be used to construct a perfectly hiding commitment schemes. Recently, Haitner et. al. [HHK$^+$05] reduced the assumption further by constructing statistically hiding commitment based on regular one-way functions with known preimage size, and more generally on one-way functions where the preimage sizes can be efficiently approximated and also on the so called approximable-size one-way functions. In their recent breakthrough result, Nguyen et al. [NOV06] show how to construct statistical zero-knowledge arguments for NP based on any one-way function. The question of whether one-way functions imply statistical commitments, however, was left open.

3.1.1 Our Results

In this chapter, we resolve the complexity of statistically hiding commitments.

THEOREM 3.1.1
If one-way functions exist, then statistically hiding commitment schemes exist.

By Impagliazzo and Luby [IL89], the existence of commitment schemes implies the existence of one-way functions and thus the above result is tight.

3.1.2 Our Techniques

Our protocol combines, in a sense, the following two cryptographic primitives: two-phase commitment schemes recently presented by Nguyen et al. [NOV06] (extending a similar notion given in [NV06]) and universal one-way hash functions presented by Naor and Yung [NY89]. Following is an informal description of the primitives (a formal definition appears in Section 3.2).

Universal one-way hash functions (UOWHFs) Universal one-way hash functions are a relaxation of the notion of collision-resistant hash functions. A family of compressing hash functions is universal one-way if no efficient algorithm succeeds in the following game with more than negligible probability. The algorithm should first announce a value x. Then, on a uniformly selected hash function f (given to the algorithm *after* it announces x), it should find $x' \neq x$ such that $f(x') = f(x)$.

Rompel [Rom90] shows that the existence of one-way functions implies the existence of universal one-way hash functions, this result was recently rewritten by Katz and Koo [KK05], adding missing details and fixing some errors.

Two-phase commitments In a two-phase commitment scheme, the sender and the receiver interact in two consecutive phases. In each phase they carry out a commitment protocol (the commit stage and the reveal stage). The transcript of the first phase is used as input for the second-phase commitment. A two-phase commitment is statistically hiding, if before each of the reveal stages the receiver has no information about the committed value. A two-phase commitment is $\binom{2}{1}$-binding, if the sender cannot cheat both in the first phase and in the second phase. Specifically, after the first-phase commit, there is a *single* value such that if the sender decommits to any other value, then the second commitment is guaranteed to be binding (in the standard sense).

Nguyen et al. [NOV06] prove that the existence of one-way functions implies some non-uniform version of two-phase commitment schemes.

The construction idea. We would like to use a two-phase commitment schemes to construct a (standard) commitment scheme. A naive attempt to design the commitment scheme may go as follows: First, the sender commits to some random string x using the first-phase commit stage. Then, the receiver flips a coin $phase \in \{\mathsf{first}, \mathsf{second}\}$, if $phase = \mathsf{first}$ then the first-phase commitment is used as the commitment (e.g., the sender sends to the receiver the exclusive-or of its secret with x). Otherwise ($phase = \mathsf{second}$), the two parties execute the first phase reveal stage and if successful (i.e., the receiver does not reject), they use the second-phase commitment (invoked with the transcript of the first-phase as input) as the commitment.

The intuition is that since the commitment is $\binom{2}{1}$-binding, the sender cannot cheat in both phases together and thus the receiver would catch a cheating sender with probability half. The problem is, however, that the sender can decide in which commitment he likes to cheat *after* knowing the value of *phase*. Hence, the sender can cheat successfully in both cases without violating the $\binom{2}{1}$-binding of the underlying protocol.

Our additional idea is to use UOWHFs in order to force the sender to decide in which phase it is about to cheat *before* knowing the value of *phase*. Our implementation is as follows: After the first-phase commit stage, the receiver selects a random (universal one-way) hash function f and the sender sends back $y = f(x)$. The protocol proceeds essentially as the naive protocol above, where any time the first-phase reveal stage is executed in the naive protocol revealing the value x' (either in the commit-stage for $phase = \mathsf{first}$ or in the reveal stage for $phase = \mathsf{second}$), the receiver also verifies that $f(x') = y$.

Assuming the hash function f is sufficiently compressing, the string x remains quite unpredictable even though $f(x)$ is sent to R (in the new variant of the protocol). Thus, in the case that $phase = \mathsf{first}$, we can still use the "entropy" remaining in x to hide the sender's secret (assuming it

is sufficiently shorter than $|x| - |f(x)|$). To show the statistical hiding in the complementary case when $phase = \mathsf{second}$, it is sufficient to note that sending $f(x)$, does not compromise the hiding property of the second-phase commitment. All in all, the protocol is statistically hiding for both choices of *phase* and thus it is statistically hiding.

To argue about the binding of the protocol, recall that the 1-out-of-2-binding property informally states that with high probability after the first-phase commit stage, there exists a *single* value $\widetilde{x}$ that allows the sender to cheat in the second-phase commitment. Now, if the sender sends y such that $f(\widetilde{x}) = y$, then in order to cheat in the case $phase = \mathsf{first}$, it will have to open the first-phase commitment to a value $x' \neq \widetilde{x}$ such that $f(x') = y = f(\widetilde{x})$. This would imply the breaking of the universal one-way hash function. On the other hand, if $f(\widetilde{x}) \neq y$, then in the case $phase = \mathsf{second}$ the sender is forced to open the first-phase commitment to a value different than $\widetilde{x}$. This guarantees that the sender cannot cheat in the second-phase commitment and thus in this case our protocol is binding. In conclusion, since y is sent before *phase* is chosen, we are guaranteed that our protocol is weakly binding (since intuitively there always exists a choice of *phase* that prevent the sender from cheating). We complete the construction by amplifying the above protocol into a full-fledged statistically hiding commitment scheme using standard techniques.

3.1.3 Outline

Section 3.2 includes the additional definitions and notations used throughout this chapter. In Section 3.3, we present our construction of statistically hiding commitment given two-phase commitment and family of universal hash functions, where in Section 3.3 we use the result of Section 3.3 to prove our main result. Discussion and further issues appear in Section 3.5.

3.2 Preliminaries

3.2.1 Universal One-way Hash Family

In order to define a universal one-way hash family, we need to understand what it means for a family of functions to be *polynomial-time computable*.

DEFINITION 3.2.1
A family of functions $\mathcal{F} = \bigcup_n \mathcal{F}_n = \{f\colon \{0,1\}^n \to \{0,1\}^{\ell(n)}\}$ is *polynomial-time computable* if

- Every function $f \in \mathcal{F}_n$ is described by a string of length $p(n)$ for some polynomial p. By abuse of notation, we also denote this description by f, and write $f \leftarrow \mathcal{F}_n$ to mean that it is chosen uniformly at random in $\{0,1\}^{p(n)}$. (A more general definition would allow the description of the function to be selected according to any polynomial-time samplable distribution, even one that requires private coin tosses. However, our stronger 'public-coin' definition is achieved by existing constructions, and can be useful in applications, such as constructing public-coin zero-knowledge arguments.

- There exists a deterministic polynomial-time algorithm F such that for every n and every $f \in \mathcal{F}_n$, given the description of the function f and a string $x \in \{0,1\}^n$, F outputs the value of $f(x)$.

DEFINITION 3.2.2
A polynomial-time computable family of functions $\mathcal{F} = \bigcup_n \mathcal{F}_n = \{f\colon \{0,1\}^n \to \{0,1\}^{\ell(n)}\}$ is a *universal one-way hash family (UOWHF)* if $\ell(n) < n$ and the for all PPT A the following is negligible in n

$$\Pr[(x, \mathsf{state}) \leftarrow A(1^n), f \leftarrow \mathcal{F}_n, x' \leftarrow A(x, \mathsf{state}, f) : x' \neq x \bigwedge f(x') = f(x)].$$

REMARK 3.2.3

- In the above definition, we allow the adversary to transfer additional information, i.e., state, between the selection of x and finding the collision. This state variable does not appear in the definition in Katz and Koo [KK05], which is otherwise identical to the above. However, any universal one-way hash family $\mathcal{F}$ meeting their weaker definition can be converted into one meeting the above definition by selecting $f \leftarrow \mathcal{F}$, $s \leftarrow \{0,1\}^n$, and defining $f'(x) = f(x \oplus s)$. (Intuitively, the random shift s turns an arbitrary point x selected by the adversary into a uniformly random point out of the adversary's control.)

  The original definition of Naor and Yung [NY89] (also used by Rompel [Rom90]) does not involve the adversary before f is chosen at all, but rather requires that for *all* $x \in \{0,1\}^n$, $A(x, f)$ has a low probability of producing a collision (over the choice of f and A's coin tosses). Their definition is suited for the case of nonuniform security (as the arbitrary x can be viewed as nonuniform advice), in which case it becomes equivalent to ours (since A can also have state hardwired nonuniformly).

- Although it is more natural for the security be parameterized in terms of the output length, namely $\ell(n)$, our applications do not require hash functions that are shrinking by more than a polynomial factor. Hence for this reason, and in part for consistency, we keep n as our security parameter.

- Naor and Yung [NY89] showed that starting with a universal one-way hash family that is compressing by only one bit, namely $\ell(n) = n - 1$, more compression can be achieved, say $\ell(n) \leq n/2$, by iterative application several hash functions chosen from the family. Moreover, it is easy to verify that the same construction holds also w.r.t. to Definition 3.2.2. Hence, without loss of generality, we can assume that our universal one-way hash family will have the feature that $\ell(n) \leq n/2$.

Two properties of a universal one-way hash family. A universal one-way hash family satisfying Definition 3.2.2 has the following two main properties.

Large preimages: most of the preimages have a large size. This follows from the compressing nature of hash functions: the output length $\ell(n)$ is much shorter than the input length n. (Recall that we can get a universal one-way hash family with $\ell(n) \leq n/2$.) We formalize this in property in Definition 3.2.4.

Target collision resistance: it is hard to find collisions with a value x announced *before* the hash function is given. We formalize this in property in Definition 3.2.5

DEFINITION 3.2.4
A family of functions $\mathcal{F} = \bigcup_n \mathcal{F}_n = \{f\colon \{0,1\}^n \to \{0,1\}^{\{0,1\}^{\ell(n)}}\}$ has the large preimages property if for every $f \in \mathcal{F}$, most elements in the range of f have large preimage sizes. Stated precisely, there exists a function $\alpha(n) = \omega(1)$ and a negligible function ε, such that for all values of n, the following holds:

$$\Pr_{x \leftarrow \{0,1\}^n}\left[\left|f^{-1}(f(x))\right| \geq n^{\alpha(n)}\right] \geq 1 - \varepsilon(n) \ ,$$

for every function $f \in \mathcal{F}_n$.

DEFINITION 3.2.5
A family of functions $\mathcal{F} = \bigcup_n \mathcal{F}_n = \{f\colon \{0,1\}^n \to \{0,1\}^{\ell(n)}\}$ has the statistical [resp., computational] target collision resistance property if for every [resp., every PPT] A, the following is negligible in n:

$$\Pr[(x, \mathsf{state}) \leftarrow A(1^n), f \leftarrow \mathcal{F}_n, x' \leftarrow A(x, \mathsf{state}, f) : x' \neq x \bigwedge f(x') = f(x)] \ .$$

Large preimages and target collision resistance are opposing properties. Specifically, it is impossible for a *single* family of functions to have large preimages and have *statistical* target collision resistance. The power of a universal one-way hash family comes from the fact that it has the large preimages property and has *computational* target collision resistance.

LEMMA 3.2.6
If $\mathcal{F} = \bigcup_n \mathcal{F}_n = \{f\colon \{0,1\}^n \to \{0,1\}^{\ell(n)}\}$, for $\ell(n) \leq n/2$, is a universal one-way hash family, then $\mathcal{F}$ has both the large preimages and the *computational* target collision resistance properties.

Proof. The computational target collision resistance property follow directly from Definition 3.2.2. Hence, all we need to show is that the compressing nature of $\mathcal{F}$, when $\ell(n) \leq n/2$, implies the large preimages property.

Group the elements with small preimages into a set $S = \{y \in \{0,1\}^{\ell(n)} : \left|f^{-1}(y)\right| < 2^{\frac{3}{4}n-\ell(n)}\}$. Since $\ell(n) \leq n/2$, every element $y \notin S$ has a preimage of size $\left|f^{-1}(y)\right| \geq 2^{\frac{3}{4}n-\ell(n)} \geq 2^{n/4} = n^{\omega(1)}$. To complete, we bound the probability of landing in S, which we do by a union bound over the elements in S (for which, there are at most $2^{\ell(n)}$):

$$\Pr_{x \leftarrow \{0,1\}^n}[f(x) \in S] = \Pr\left[\exists y \in S \text{ with } f(U_n) = y\right] < \frac{2^{\frac{3}{4}n-\ell(n)}}{2^n} \cdot 2^{\ell(n)} = 2^{-n/4} = \mathrm{neg}(n) \ . \qquad \square$$

3.2.2 Commitment Schemes

Another basic primitive of modern cryptography is a *(bit) commitment scheme*, which is a two-stage protocol between a sender and a receiver. In the first stage, called the *commit stage*, the sender *commits* to a private bit b. In the second stage, called the *reveal stage*, the sender reveals b and *proves* that it was the bit to which she committed in the first stage. We require two properties of commitment schemes. The *hiding* property says that the receiver learns nothing about b in the commit stage. The *binding* property says that after the commit stage, the sender is bound to a particular value of b; that is, she cannot successfully open the commitment to two different bits in the reveal stage.

DEFINITION 3.2.7 (commitment schemes)
An commitment scheme is an interactive protocol $\mathsf{Com} = (\mathsf{S}, \mathsf{R})$ with the following properties:

1. Scheme Com proceeds in two stages: a commit stage and a reveal stage. In both stages, the sender S and the receiver R receive a security parameter 1^n as common input.
2. At the beginning of the commit stage, sender S receives a private input $b \in \{0,1\}$, which denotes the bit that S is supposed to commit to. The commitment stage results in a joint output, which we call the commitment $c = \text{output}((\mathsf{S}(b), \mathsf{R})(1^n))$, and a private output for S, which we call the decommitment string $d = \text{output}_{\mathsf{S}}(\mathsf{S}(b), \mathsf{R})(1^n)$. Without loss of generality, c can be taken to be the full transcript of the interaction between S and R, and d to be the private coin tosses of S.
3. In the reveal stage, sender S sends the pair (b, d), where d is the decommitment string for bit b. Receiver R accepts or rejects based on b, d, and c.
4. The sender S and receiver R algorithms are computable in polynomial time in the security parameter n.
5. R will always accept (with probability 1) if both sender S and receiver R follow their prescribed strategy.

A commitment scheme is public coin if all messages sent by the receiver are independent random coins.

Next, we define the hiding and binding properties of commitment schemes.

DEFINITION 3.2.8 (hiding)
Commitment scheme $\mathsf{Com} = (\mathsf{S}, \mathsf{R})$ is statistically [resp., computationally] hiding if for every [resp., PPT] R^*, the ensembles $\{\text{view}_{\mathsf{R}^*}(\mathsf{S}(0), \mathsf{R}^*)(1^n)\}_{n\in\mathbb{N}}$ and $\{\text{view}_{\mathsf{R}^*}(\mathsf{S}(1), \mathsf{R}^*)(1^n)\}_{n\in\mathbb{N}}$ are statistically [resp., computationally] indistinguishable, where $\text{view}_{\mathsf{R}^*}(\mathsf{S}(b), \mathsf{R}^*)$ denotes the view of R^* in the commit stage interacting with $\mathsf{S}(b)$.

DEFINITION 3.2.9 (binding)
Commitment scheme $\mathsf{Com} = (\mathsf{S}, \mathsf{R})$ is statistically [resp., computationally] binding if for every [resp., PPT] S^*, there exists a negligible function ε such that the malicious sender S^* succeeds in the following game with probability at most $\varepsilon(n)$:

> On security parameter 1^n, S^* interacts with R in the commit stage obtaining commitment c. Then S^* outputs pairs $(0, d_0)$ and $(1, d_1)$, and *succeeds* if in the reveal stage, $\mathsf{R}(0, d_0, c) = \mathsf{R}(1, d_1, c) = \texttt{accept}$.

If the above holds for every nonuniform PPT S^*, we say that Com is computationally binding with nonuniform security.

Constructing commitment schemes based on any one-way function. Naor [Nao91] constructed commitment schemes that are computationally hiding and statistically binding from any *pseudorandom generator*, which in turn can be based on any one-way function [HILL99]. The main result of this chapter, Theorem 3.1.1, shows that commitments schemes that are *statistically hiding* and computationally binding commitments can be based on any one-way function.

3.2.3 Two-phase Commitment Schemes

Two-phase commitment schemes are an alternate variant of commitments introduced by Nguyen et al. [NOV06] (extending a similar notion given in Nguyen and Vadhan [NV06]). These are commitment schemes with two *sequential* and *related* stages such that in each stage, the sender commits to and reveals a value.

DEFINITION 3.2.10 (two-phase commitment schemes)
A two-phase commitment scheme (S, R), with security parameter n and message lengths $(k_1(n), k_2(n))$, consists of four interactive protocols: the first commitment stage $(\mathsf{S}_c^1, \mathsf{R}_c^1)$, the first reveal stage $(\mathsf{S}_r^1, \mathsf{R}_r^1)$, the second commitment stage $(\mathsf{S}_c^2, \mathsf{R}_c^2)$, and the second reveal stage $(\mathsf{S}_r^2, \mathsf{R}_r^2)$. For us, both reveal phases will always be noninteractive, consisting of a single message from the sender to the receiver.

1. In the first commitment stage, S_c^1 receives a private input $\sigma^{(1)} \in \{0,1\}^{k_1}$ and coin tosses r_{S}. At the end of the interaction, both S_c^1 and R_c^1 output a commitment $c^{(1)}$. (Without loss of generality, we can assume that $c^{(1)}$ is the transcript of the first commitment stage.)

2. In the first (noninteractive) reveal stage, both S_r^1 and R_r^1 receive as common inputs the commitment $c^{(1)}$, and S_r^1 receives as private input its previous coin tosses r_{S}. S_r^1 sends R_r^1 a pair $(\sigma^{(1)}, \gamma^{(1)})$ with $\gamma^{(1)}$ interpreted as a decommitment for $\sigma^{(1)} \in \{0,1\}^{k_1}$. R_r^1 accepts or rejects based on $c^{(1)}$, $\sigma^{(1)}$, and $\gamma^{(1)}$. After that, both S_r^1 and R_r^1 outputs a string τ. (Without loss of generality, we can assume that τ is the transcript of the first commitment stage and the first reveal stage and includes R_r^1's decision to accept or reject.)

3. In the second commitment stage, both S_c^2 and R_c^2 receive as common input the string τ, and S_c^2 receives a private input $\sigma^{(2)} \in \{0,1\}^{k_2}$ and its previous coin tosses r_{S}. At the end of the interaction, both S_c^2 and R_c^2 output a commitment $c^{(2)}$. (Without loss of generality, we can assume that $c^{(2)}$ is the concatenation of τ and the transcript of the second commitment stage.)

4. In the second (noninteractive) reveal stage, both S_r^2 and R_r^2 receive as common input the commitment $c^{(2)}$, and S_r^2 receives as private input its previous coin tosses r_{S}. S_r^2 sends R_r^2 a pair $(\sigma^{(2)}, \gamma^{(2)})$ with $\gamma^{(2)}$ interpreted as a decommitment for $\sigma^{(2)} \in \{0,1\}^{k_2}$. R_r^2 accepts or rejects based on $c^{(2)}$, $\sigma^{(2)}$, and $\gamma^{(2)}$.

- We insist that scheme (S, R) have *perfect completeness*. That is to say, if both sender S and receiver R follow their prescribed strategy, then R will always accept (with probability 1).
- The sender and receiver's algorithms, denoted by $\mathsf{S} = (\mathsf{S}^1, \mathsf{S}^2) = ((\mathsf{S}_c^1, \mathsf{S}_r^1), (\mathsf{S}_c^2, \mathsf{S}_r^2))$ and $\mathsf{R} = (\mathsf{R}^1, \mathsf{R}^2) = ((\mathsf{R}_c^1, \mathsf{R}_r^1), (\mathsf{R}_c^2, \mathsf{R}_r^2))$ respectively, are computable in polynomial time.
- Scheme (S, R) is *public coin* if all messages sent by R to S are independent random coins.

Hiding for two-phase commitment schemes. As for standard commitment schemes, we define the security of the sender in terms of a hiding property. Stated informally, the hiding property for a two-phase commitment scheme says that *both* commitment phases are hiding. Note that since the phases are run sequentially, the hiding property for the second commitment stage is required to hold even given the receiver's view of the first stage.

DEFINITION 3.2.11
Two-phase commitment scheme (S, R), with security parameter n and message lengths $(k_1(n), k_2(n))$, is statistically hiding if for all adversarial receiver R^*,

1. The views of R^* when interacting with the sender in the first phase on any two messages are statistically indistinguishable. Namely, for all $\sigma^{(1)}, \widetilde{\sigma}^{(1)} \in \{0,1\}^{k_1}$, the probability ensembles $\left\{\text{view}_{\mathsf{R}^*}(\mathsf{S}_c^1(\sigma^{(1)}), \mathsf{R}^*)(1^n)\right\}_{n\in\mathbb{N}}$ and $\left\{\text{view}_{\mathsf{R}^*}(\mathsf{S}_c^1(\widetilde{\sigma}^{(1)}), \mathsf{R}^*)(1^n)\right\}_{n\in\mathbb{N}}$ are statistically indistinguishable.

2. The views of R^* when interacting with the sender in the second phase are statistically indistinguishable no matter what the sender committed to in the first phase. Namely, for all $\sigma^{(1)} \in \{0,1\}^{k_1}$, and all $\sigma^{(2)}, \widetilde{\sigma}^{(2)} \in \{0,1\}^{k_2}$, the probability ensembles $\left\{\text{view}_{\mathsf{R}^*}(\mathsf{S}_c^2(\sigma^{(2)}), \mathsf{R}^*)(\mathrm{T}, 1^n)\right\}_{n\in\mathbb{N}}$ and $\left\{\text{view}_{\mathsf{R}^*}(\mathsf{S}_c^2(\widetilde{\sigma}^{(2)}), \mathsf{R}^*)(\mathrm{T}, 1^n)\right\}_{n\in\mathbb{N}}$, where $\mathrm{T} = \mathsf{tra}(\mathsf{S}^1(\sigma^{(1)}), \mathsf{R}^*)(1^n)$, are statistically indistinguishable.

We stress that the second condition of the above hiding definition (Definition 3.2.11) requires that the view of receiver in the second phase be indistinguishable for any two messages even given the transcript of the first phase, $\mathrm{T} = \mathsf{tra}(\mathsf{S}^1(\sigma^{(1)}), \mathsf{R}^*)(1^n)$.

1-out-of-2 binding for two-phase commitment schemes. The 1-out-of-2 binding property, informally stated, says that *at least* one of the two commitment phases is binding. In other words, for every PPT malicious sender S^*, at most one of the two phases is bad in that S^* can decommit a given commitment to two different messages in that phase. We allow this bad phase to be determined dynamically by S^*. Moreover, we require that the second phase be *statistically* binding if the sender breaks the first phase. Our construction achieves this stronger property, and using it simplifies some of our proofs.

DEFINITION 3.2.12
Two-phase commitment scheme (S, R), with security parameter n and message lengths $(k_1(n), k_2(n))$, is computationally 1-out-of-2 binding if there exists a set $\mathcal{B}$ of first phase transcripts such that for every function $\varepsilon(n) = 1/\operatorname{poly}(n)$, the following holds:

1. For all PPT adversary S^*, S^* succeeds in the following game with probability at most $\varepsilon(n)$ for all sufficiently large n:
   (a) S^* and R_c^1 interact and output a first-phase commitment $c^{(1)}$.
   (b) S^* outputs two full transcripts $\lambda = (\tau, \kappa)$ and $\widetilde{\lambda} = (\widetilde{\tau}, \widetilde{\kappa})$ of *both* phases with the following three properties:
      - Transcripts λ and $\widetilde{\lambda}$ both start with prefix $c^{(1)}$.

- Transcript λ contains a successful opening of $c^{(1)}$ to the value $\sigma^{(1)} \in \{0,1\}^{k_1}$ using a first-phase transcript τ not in $\mathcal{B}$, and R_r^1 and R_r^2 both accept in λ.
- Transcript $\widetilde{\lambda}$ contains a successful opening of $c^{(1)}$ to the value $\widetilde{\sigma}^{(1)} \in \{0,1\}^{k_1}$ using a first-phase transcript $\widetilde{\tau}$ not in $\mathcal{B}$, and R_r^1 and R_r^2 both accept in $\widetilde{\lambda}$.

(c) S^* succeeds if all of the above conditions hold and $\sigma^{(1)} \neq \widetilde{\sigma}^{(1)}$.

2. For every (even computationally unbounded) sender S^*, the first-phase transcripts in $\mathcal{B}$ make the second phase statistically binding. In other words, for all S^*, all $\tau \in \mathcal{B}$, and all sufficiently large n, with probability at least $1 - \varepsilon(n)$ over $c^{(2)} = (\mathsf{S}^*, \mathsf{R}_c^2)(\tau)$, there is at most one value $\sigma^{(2)} \in \{0,1\}^{k_2}$ such that $\text{output}_{\mathsf{R}}(\mathsf{S}^*, \mathsf{R}_r^2)(c^{(2)}, \sigma^{(2)}) = \texttt{accept}$.

The following theorem was proven in Nguyen et al. [NOV06].

THEOREM 3.2.13

If one way functions exist, then there exists an efficient procedure that on security parameter n, outputs a collection of public-coin two-phase commitment schemes $\mathcal{COM} = \{\mathsf{Com}_1, \cdots, \mathsf{Com}_m\}$ with message lengths $(k_1, k_2) = (n, n)$, such that:

- there exists an index $i \in \{1, 2, \ldots, m\}$ such that scheme Com_i is statistically hiding, and
- for every index $i \in \{1, 2, \ldots, m\}$, scheme Com_i is computationally 1-out-of-2 binding.

The statistical hiding property above holds regardless of whether or not f is secure (hard to invert). On the other hand if f is nonuniformly secure, then the 1-out-of-2 binding above will be with nonuniform security.

3.3 The Construction

We present the transformation algorithm using two-phase commitment scheme and an arbitrary family of functions $\mathcal{F}$, and will only require $\mathcal{F}$ to be a universal one-way hash family when we want to prove the hiding and binding security properties.

ALGORITHM 3.3.1

The transformation, denoted as 2-to-1-Transform.

Input: security parameter 1^n, two-phase commitment scheme (S, R) with message lengths $(k_1, k_2) = (n, 1)$, and a family of functions $\mathcal{F} = \bigcup_n \mathcal{F}_n = \{f : \{0,1\}^n \to \{0,1\}^{\ell(n)}\}$.

Output: Commitment scheme $(\mathbb{S}, \mathbb{R})$ as described by Protocol 3.3.2.

Hence, we write the commitment scheme obtained as $(\mathbb{S}, \mathbb{R}) = \text{2-to-1-Transform}((\mathsf{S}, \mathsf{R}), \mathcal{F})$.

PROTOCOL 3.3.2

Standard commitment scheme $(\mathbb{S}, \mathbb{R})$ from two-phase commitment scheme (S, R).

Security parameter: 1^n, given as common input to both $\mathbb{S}$ and $\mathbb{R}$.

Sender's private input: Bit $b \in \{0,1\}$.

Commit stage:

1. $\mathbb{S}$ selects a uniform $\sigma \leftarrow \{0,1\}^n$.
2. $\mathbb{S}$ and $\mathbb{R}$ engage in $(\mathsf{S}_c^1(\sigma), \mathsf{R}_c^1)(1^n)$, with $\mathbb{S}$ acting as S_c^1 and $\mathbb{R}$ acting as R_c^1. Let $c^{(1)}$ be the common output of S_c^1 and R_c^1 after the interaction.
3. $\mathbb{R}$ chooses $f \leftarrow \mathcal{F}_n$ and sends it to $\mathbb{S}$.
4. $\mathbb{S}$ sends $y = f(\sigma)$ to $\mathbb{R}$.
5. $\mathbb{R}$ flips a random coin, represented by $phase \leftarrow \{1,2\}$, and sends $phase$ to $\mathbb{S}$.
   If $phase = 1$, then proceed as follows:
   (a) $\mathbb{S}$ selects a random hash $h \leftarrow \mathcal{H}$, where $\mathcal{H}$ is a family of pairwise-independent hash functions with domain $\{0,1\}^n$ and range $\{0,1\}$, and sends $(h, b \oplus h(\sigma))$ to $\mathbb{R}$.
   (b) $\mathbb{S}$ and $\mathbb{R}$ both output $(c^{(1)}, f, y, phase = 1, h, b \oplus h(\sigma))$ as the commitment.
   If $phase = 2$, then proceed as follows:
   (a) $\mathbb{S}$ runs S_r^1 to obtain the decommitment message $\gamma^{(1)}$ and first-phase transcript τ corresponding to both σ and $c^{(1)}$. $\mathbb{S}$ sends $(\sigma, \gamma^{(1)}, \tau)$ to $\mathbb{R}$.
   (b) $\mathbb{S}$ and $\mathbb{R}$ engage in $(\mathsf{S}_c^2(b), \mathsf{R}_c^2)(1^n, \tau)$, with $\mathbb{S}$ acting as S_c^2 and $\mathbb{R}$ acting as R_c^2. Let $c^{(2)}$ be the common output of S_c^2 and R_c^2 after the interaction.
   (c) $\mathbb{S}$ and $\mathbb{R}$ both output $(c^{(1)}, f, y, phase = 2, c^{(2)})$ as the commitment.

Reveal stage:

To decommit to bit b, do the following depending the value of $phase$.

If $phase = 1$, then:

1. $\mathbb{S}$ sends (b, σ) to $\mathbb{R}$;
2. If $y = f(\sigma)$ and the last component of the commitment equals $b \oplus h(\sigma)$, then $\mathbb{R}$ *accepts*. Otherwise, $\mathbb{R}$ *rejects*.

If $phase = 2$, then:

1. $\mathbb{S}$ runs S_r^2 to obtain the decommitment message $\gamma^{(2)}$, and sends $(b, \gamma^{(2)})$ to $\mathbb{R}$;
2. If $y = f(\sigma)$ and both R_r^1 and R_r^2 accept $(c^{(1)}, \sigma, \gamma^{(1)})$ and $(c^{(2)}, b, \gamma^{(2)})$, respectively, then $\mathbb{R}$ *accepts*. Otherwise, $\mathbb{R}$ *rejects*.

3.3.1 Analyzing the Transformation

The hiding and binding security properties of Protocol 3.3.2 will rely on properties of $\mathcal{F}$ being a universal one-way hash family.

Hiding

We now show that the large preimages property of $\mathcal{F}$ (see Lemma 3.2.6) translates to the hiding property of the commitment scheme $(\mathbb{S}, \mathbb{R}) = \text{2-to-1-}\mathsf{Transform}((\mathsf{S}, \mathsf{R}), \mathcal{F})$.

LEMMA 3.3.3
If the family of functions $\mathcal{F}$ has the large preimages property, and the two-phase commitment scheme (S, R) is statistically hiding, then scheme $(\mathbb{S}, \mathbb{R}) = \text{2-to-1-}\mathsf{Transform}((\mathsf{S}, \mathsf{R}), \mathcal{F})$ is statistically hiding.

Proof. What we need to show is that for any adversarial receiver R^*, the views of R^* in $(\mathbb{S}(0), \mathsf{R}^*)$ and $(\mathbb{S}(1), \mathsf{R}^*)$ are statistically indistinguishable. (In this proof, we drop the security parametrization of 1^n because it is clear from context.) We can, without loss of generality, only consider deterministic R^* because we can fix the adversary's coin tosses to maximize its distinguishing advantage. In the rest of this proof, we use *indistinguishability* and *hiding* to mean those of the statistical variant.

Let P denote the value of *phase* sent by R^*, and we break our hiding analysis to cases when $P = 1$ and $P = 2$. To formalize this case analysis, we say that random variables X and Y are ***indistinguishable on event*** E if for all D, $|\Pr[D(X) = 1 \wedge E] - \Pr[D(X) = 0 \wedge E]|$ is negligible (in the security parameter n). What we will show is that the random variables $\text{view}_{\mathsf{R}^*}(\mathbb{S}(0), \mathsf{R}^*)$ and $\text{view}_{\mathsf{R}^*}(\mathbb{S}(1), \mathsf{R}^*)$ are indistinguishable on both events $P = 1$ and $P = 2$, thus allowing us to conclude that the scheme is hiding.

First, we analyze the case when $P = 2$. Let the random variables Σ and F denote $\mathbb{S}$'s choice of σ and the value of f sent by R^*, respectively. Observe that P is a *deterministic* function of the random variables $V_1 = \text{view}_{\mathsf{R}^*}(\mathsf{S}_c^1(\Sigma), \mathsf{R}^*)$ and $Y = F(\Sigma)$. In turn, V_1 and Y are deterministic functions of the first-phase transcript $\mathrm{T} = \mathsf{trans}(\mathsf{S}^1(\Sigma), \mathsf{R}^*)$, which includes both the commit and reveal stages. This is because we can compute the view of the receiver from the first-phase transcript, and the first-phase transcript also contains the value of σ, from which we can compute $y = f(\sigma)$. For bit $b \in \{0, 1\}$, let random variable $V_2(b) = \text{view}_{\mathsf{R}^*}(\mathsf{S}_c^2(b), \mathsf{R}^*)(\mathrm{T})$, recalling that $\mathrm{T} = \mathsf{trans}(\mathsf{S}^1(\Sigma), \mathsf{R}^*)$. Because (S, R) is hiding, its two-phase commitments is hiding even given the first-phase transcript: this means that $(V_2(0), \mathrm{T})$ is indistinguishable from $(V_2(1), \mathrm{T})$. Since P is a deterministic function of T, random variables $(V_2(0), \mathrm{T})$ and $(V_2(1), \mathrm{T})$ are indistinguishable on event $P = 2$. Since $\text{view}_{\mathsf{R}^*}(\mathbb{S}(b), \mathsf{R}^*)|_{P=2}$ is a deterministic function of $(V_2(b), \mathrm{T})|_{P=2}$, for $b \in \{0, 1\}$, we have that $\text{view}_{\mathsf{R}^*}(\mathbb{S}(0), \mathsf{R}^*)$ and $\text{view}_{\mathsf{R}^*}(\mathbb{S}(1), \mathsf{R}^*)$ are indistinguishable on event $P = 2$.

Next, we analyze the case when $P = 1$. The hiding property of the first phase gives us

$$(V_1, \Sigma) \approx_s (V_1, U_n) \ ,$$

where U_n represent a uniform random variable over $\{0, 1\}^n$, and is independent from V_1 and Σ. Recall that the random variable F denotes the function f sent by R^*. Since F is a deterministic function of V_1, we get

$$(V_1, F, F(\Sigma), \Sigma) \approx_s (V_1, F, F(U_n), U_n) \ .$$

Now, let the random variable H represent the hash function h selected by $\mathbb{S}$ when *phase* $= 1$. Note that H is independent of V_1, F, Σ, and U_n, so

$$(V_1, F, Y, H, H(\Sigma)) \approx_s (V_1, F, F(U_n), H, H(U_n)) \ , \tag{3.1}$$

recalling that $Y = F(\Sigma)$.

What we need to establish is that $H(U_n)$ is close to uniform so that we have hiding. The next claim does this for us.

CLAIM 3.3.4
Suppose family of functions $\mathcal{F} = \bigcup_n \mathcal{F}_n$ has the large preimages property. Let the random variable H denote a random hash function from a family of pairwise-independent hash functions with domain $\{0,1\}^n$ and range $\{0,1\}$, random variable U_n denote a uniform string in $\{0,1\}^n$, random variable U_1' denote a uniform string in $\{0,1\}$, and that H, U_n, and U_1' are all independent. For every $f \in \mathcal{F}_n$, $(f(U_n), H, H(U_n))$ is indistinguishable from $(f(U_n), H, U_1')$.

Proof. The large preimages property of $\mathcal{F}$ guarantees that with probability $1 - \mathrm{neg}(n)$ over $y \leftarrow f(U_n)$, the min-entropy $\mathrm{H}_\infty(U_n|_{f(U_n)=y}) \geq \omega(\log n)$. For y satisfying this condition, we apply the Leftover Hash Lemma 2.2.4 to get that $(y, H, H(U_n|_{f(U_n)=y}))$ is indistinguishable from $(y, H, H(U_n|_{f(U_n)=y}))$. □

Because H and U_n are independent from the rest of the random variables (and are independent from each other), Claim 3.3.4 states that

$$(V_1, F, F(U_n), H, H(U_n)) \approx_s (V_1, F, F(U_n), H, U_1') \ , \tag{3.2}$$

where U_1' is an independent random variable representing a uniform random variable over $\{0,1\}$. Combining (3.1) and (3.2), we get

$$(V_1, F, Y, H, H(\Sigma)) \approx_s (V_1, F, F(U_n), H, U_1') \ ,$$

which leads to:

$$\begin{aligned}(V_1, F, Y, H, 0 \oplus H(\Sigma)) &\approx_s (V_1, F, F(U_n), H, 0 \oplus U_1') \\ &\equiv (V_1, F, F(U_n), H, 1 \oplus U_1') \\ &\approx_s (V_1, F, Y, H, 1 \oplus H(\Sigma)) \ .\end{aligned}$$

Since P is a deterministic function of V_1 and Y, random variables $(V_1, F, Y, H, 0 \oplus H(\Sigma))$ and $(V_1, F, Y, H, 1 \oplus H(\Sigma))$ are indistinguishable on event $P = 1$. Since $\mathrm{view}_{\mathsf{R}^*}(\mathbb{S}(b), \mathsf{R}^*)|_{P=1}$ is a deterministic function of $(V_1, F, Y, H, b \oplus H(\Sigma))|_{P=1}$, for $b \in \{0,1\}$, we have that $\mathrm{view}_{\mathsf{R}^*}(\mathbb{S}(0), \mathsf{R}^*)$ and $\mathrm{view}_{\mathsf{R}^*}(\mathbb{S}(1), \mathsf{R}^*)$ are indistinguishable on event $P = 1$. □

Binding

We show that the target collision resistance property of $\mathcal{F}$ translates to the binding property of the commitment scheme $(\mathbb{S}, \mathbb{R}) = \text{2-to-1-}\mathsf{Transform}((\mathsf{S}, \mathsf{R}), \mathcal{F})$ obtained from the 2-to-1-$\mathsf{Transform}$. Because we will only be able to show that $(\mathbb{S}, \mathbb{R})$ is binding with probability close to $1/2$, we first define what it means to for a scheme to be binding with probability δ, for some $\delta \in [0,1]$.

DEFINITION 3.3.5
Commitment scheme (S, R) is statistically [resp. computationally] $\delta(n)$-binding if for every [resp. every PPT] S^* and every large enough values of n, sender S^* succeeds in the following game with probability at most $\delta(n)$:

On security parameter 1^n, S^* interacts with R in the commit stage obtaining commitment c. Then S^* outputs pairs $(0, d_0)$ and $(1, d_1)$, and *succeeds* if in the reveal stage, $\mathsf{R}(0, d_0, c) = \mathsf{R}(1, d_1, c) = \texttt{accept}$.

The standard notion of binding as given in Definition 3.2.9 corresponds to being statistically [resp. computationally] $1/p(n)$-binding for every polynomial p.

LEMMA 3.3.6

If the family of functions $\mathcal{F}$ is statistically [resp., computationally] target collision resistant, and the two-phase commitment scheme (S, R) is statistically [resp., computationally] $\binom{2}{1}$ binding, then the scheme $(\mathbb{S}, \mathbb{R}) = \text{2-to-1-Transform}((\mathsf{S}, \mathsf{R}), \mathcal{F})$ is statistically [resp., computationally] $(1/2 + 1/p(n))$-binding for every polynomial p and sufficiently large n.

Proof. We will focus on the case of computational binding. The statistical case will follow from the fact that the proof is "black box". Specifically, our proof will (implicitly) give efficient reductions M_1, M_2 such that given any sender strategy S^* that breaks the $(1/2 + 1/p(n))$-binding property of $(\mathbb{S}, \mathbb{R})$ as oracle, either $M_1^{\mathsf{S}^*}$ will break the target collision resistance property of $\mathcal{F}$ with non-negligible probability or $M_2^{\mathsf{S}^*}$ will break the $\binom{2}{1}$ binding property of (S, R). If both $\mathcal{F}$ and (S, R) have statistical [resp., computational] security, then this is impossible for every strategy [resp., every PPT strategy] S^* and we deduce that $(\mathbb{S}, \mathbb{R})$ must be statistically [resp., computationally] $(1/2 + 1/p(n))$-binding.

Unless stated otherwise, we take probabilities over the entire interaction between S^* and $\mathbb{R}$ in both the commit and reveal stages. We say that S^* ***succeeds*** if it is able to produce decommitments to two different messages for commitment Υ in the reveal phase (recall that, the reveal stage is non-interactive). We want to prove that $\Pr[\mathsf{S}^* \text{ succeeds}] \leq 1/2 + 1/p(n)$. We will do this by breaking the probability space into events $E_1, \ldots, E_5$ corresponding to the various cases in the intuitive proof outline given in Section 3.1.2. We will show that $\Pr[\bigvee_i E_i] = 1$, $\Pr[E_1] = 1/2$ and $\Pr[\mathsf{S}^* \text{ succeeds} \wedge E_i] \leq 1/4p(n)$ for $i = 2, \ldots, 5$, and this will suffice to prove the lemma.

The first event, E_1, will depend on the random variables $C = \text{view}_{\mathsf{S}^*}(\mathsf{S}^*, \mathsf{R}_c^1)$, representing S^*'s view of the first phase commit (this determines the entire state of the interaction $(\mathsf{S}^*, \mathsf{R})$, since by Definition 3.2.10 the honest receiver maintains no private state after the commit phase other than the commitment string); Y, denoting the hash value sent by S^* after the first-phase commit; P, representing the value of *phase*; and F, representing the choice of the function $f \leftarrow \mathcal{F}$. We would also like to consider whether or not Y equals $f(\Sigma^*)$, where Σ^* intuitively represents the value to which C is a commitment, i.e. the 'unique' value that will enable S^* to break the binding property of the 2nd phase. However, since the commitment scheme may be only computationally binding, Σ^* is not defined information-theoretically. Thus, we define it as the most likely value to which S^* will open the first-phase commitment (with a transcript not in $\mathcal{B}$). Formally, for each first-phase commit transcript $c \in \text{Supp}(C)$, we define:

$$p_\sigma[c] = \Pr\left[\begin{array}{c} (\mathsf{S}^*, \mathsf{R}) \text{ includes an } \textit{accepting} \text{ full transcript } \lambda = (\tau, \kappa) \\ \text{such that } \tau \notin \mathcal{B} \text{ and } \tau \text{ contains an opening to } \sigma \end{array} \middle| C = c \right] , \tag{3.3}$$

where we say full transcript λ is ***accepting*** if both R_r^1 and R_r^2 accept in λ. With this measure, we define $\sigma^*[c] = \text{argmax}_\sigma\, p_\sigma[c]$, breaking ties arbitrarily (say, by choosing the lexicographic smallest σ). Then we define the random variable $\Sigma^* = \sigma^*[C]$.

The intuition described in Section 3.1.2 suggests a case analysis based on whether or not $Y = F(\Sigma^*)$. According to that intuition, the scheme will be binding if $Y = F(\Sigma^*)$ and $P = 1$ (by target collision resistance of $\mathcal{F}$) or if $Y \neq F(\Sigma^*)$ and $P = 2$ (by the 1-out-of-2 binding property), and these events happen with probability $1/2$ (because P is randomly chosen after Σ^*, F, and Y are determined). This intuition can be turned directly into a proof in the case that $\mathcal{F}$ has *nonuniform* target collision resistance, since the value of Σ^* (which is determined before F) can be hardwired into the adversary breaking $\mathcal{F}$. However, to prove our result for uniform adversaries as claimed, we need to ensure that $\Sigma^* = \sigma^*[C]$ can be efficiently computed (before being given F, as per Definition 3.2.5). We observe that this is the case if $p_{\Sigma^*}[C] > 1/4p(n)$, because then if we simulate a continuation of the execution of $(\mathsf{S}^*, \mathsf{R})$ starting after C, we have a non-negligible probability of Σ^* being revealed. On the other hand the case that $p_{\Sigma^*}[C] \leq 1/4p(n)$ turns out to be analyzable similarly to the case that $Y \neq F(\Sigma^*)$; in both cases we simply use the fact that S^* is unlikely to produce a successful opening to Σ^*.

With the above in mind, we begin by analyzing the event in which we do not expect the scheme to be binding.

CLAIM 3.3.7
For the event

$$E_1 = \left\{ \begin{array}{c} [[(Y = F(\Sigma^*)) \wedge (p_{\Sigma^*}[C] > 1/4p(n))] \wedge [P = 2]] \\ \vee [[(Y \neq F(\Sigma^*)) \vee (p_{\Sigma^*}[C] \leq 1/4p(n))] \wedge [P = 1]] \end{array} \right\},$$

we have $\Pr[E_1] = 1/2$.

CLAIM 3.3.8
P is chosen randomly in $\{1, 2\}$ after C, Σ^*, F, and Y are determined.

Now we want to show that the scheme is binding on the complement of E_1. First we handle the case that $P = 1$.

CLAIM 3.3.9
For the event

$$E_2 = \{[Y = F(\Sigma^*)] \wedge [p_{\Sigma^*}[C] > 1/4p(n)] \wedge [P = 1]\},$$

we have $\Pr[\mathsf{S}^* \text{ succeeds} \wedge E_2] \leq 1/4p(n)$.

Proof. Suppose for contradiction that $\Pr[\mathsf{S}^* \text{ succeeds} \wedge E_2] > 1/4p(n)$; we will show that we can break the target collision resistance property of $\mathcal{F}$ with non-negligible probability. In order to do so, we need to output an element x before seeing the hash function, and then given a random function $f \leftarrow \mathcal{F}$, we need to output $x' \neq x$ such that $f(x) = f(x')$. We do this as follows. First we simulate the interaction between S^* and R up to the end of the first-phase commitment, and record c as the sender's view so far. Then we continue the interaction from c to the end and set x to be the value of σ sent by S^* in the protocol. (In case $phase = 1$ and S^* produces two values for σ in breaking the scheme, choose one of the two at random.) Now we output x and store $\mathsf{state} = c$,

and receive a random hash function $f \leftarrow \mathcal{F}$. We now rerun the interaction between S^* and R, starting with the view (c, f), and set x' to be the value of σ sent by S^* in the protocol (again choosing randomly if $phase = 1$ and S^* produces two values).

To see that this strategy breaks the target collision resistance property with non-negligible probability, consider the second completed execution of the interaction between S^* and R (the one with the given hash function f, which we now denote as a random variable F). By assumption, with probability greater than $1/4p(n)$ in this execution, it holds that S^* succeeds, $Y = F(\Sigma^*)$, $p_{\Sigma^*}[C] > 1/4p(n)$, and $P = 1$. Since S^* succeeds and $P = 1$, it must be the case that S^* produces two successful openings Σ_1, Σ_2 to the first-phase commit. At least one of these is different from Σ^*, yet both must satisfy $F(\Sigma_i) = Y = F(\Sigma^*)$. With probability at least $1/2$, we output $\Sigma_i \neq \Sigma^*$ as x'. Now, conditioned on all this, we argue that we had non-negligible probability (at least $(1/2) \cdot 1/4p(n)$) of outputting Σ^* as x (prior to receiving F). This follows because $p_{\Sigma^*}[C] > 1/4p(n)$. Therefore, we break the target collision resistance property with probability at least $(1/4p(n)) \cdot (1/2) \cdot (1/2) \cdot (1/4p(n))$, which is a contradiction. □

Now we turn to the complement of E_1 in case $P = 2$, namely the event

$$E' = \{[(Y \neq F(\Sigma^*)) \vee (p_{\Sigma^*}[C] \leq 1/4p(n))] \wedge [P = 2]\},$$

Since we are now restricted to $P = 2$, there is a single first-phase decommitment value produced by S^*, which we denote by the random variable Σ.

First we argue that it is almost always the case in E' that $\Sigma \neq \Sigma^*$ (assuming S^* succeeds).

CLAIM 3.3.10
For the event

$$E_3 = E' \wedge (\Sigma = \Sigma^*),$$

we have $\Pr[\mathsf{S}^* \text{ succeeds} \wedge E_3] \leq 1/4p(n)$.

Proof. In E', we either have $Y \neq F(\Sigma^*)$, in which S^* cannot succeed unless $\Sigma \neq \Sigma^*$, or we have $p_{\Sigma^*}[C] \leq (1/4(p(n)))$, in which case S^* successfully opens to value Σ^* with probability at most $1/4p(n)$. □

So now, instead of E', we can focus on the event that $\{[\Sigma \neq \Sigma^*] \wedge [P = 2]\}$. For this, we have two cases, depending on whether the transcript T of the first-phase commitment (including the reveal) gives a binding second phase or not.

CLAIM 3.3.11
For the event

$$E_4 = \{[\Sigma \neq \Sigma^*] \wedge [P = 2] \wedge [\mathrm{T} \in \mathcal{B}]\},$$

we have

$$\Pr[\mathsf{S}^* \text{ succeeds} \wedge E_4] \leq 1/4p(n).$$

Proof. If $\mathrm{T} \in \mathcal{B}$, then the second-phase commitment is binding. Since $P = 2$, S^* can only succeed with negligible probability. □

CLAIM 3.3.12

For the event

$$E_5 = \{[\Sigma \neq \Sigma^*] \wedge [P = 2] \wedge [\mathrm{T} \notin \mathcal{B}]\},$$

we have

$$\Pr[\mathsf{S}^* \text{ succeeds} \wedge E_5] \leq 1/4p(n).$$

Proof. Assume for contradiction that $\Pr[\mathsf{S}^* \text{ succeeds} \wedge E_5] > 1/4p(n)$. By Markov, this implies that with probability at least $1/8p(n)$ over $c \leftarrow C$, it holds that

$$\Pr[\mathsf{S}^* \text{ succeeds} \wedge E_5 | C = c] > 1/8p(n). \tag{3.4}$$

We will use this to break the first-phase binding of R. Similarly to the proof of Claim 3.3.9, we carry out two executions of $(\mathsf{S}^*, \mathsf{R})$ beginning with the same first-phase commit c. Assume that c satisfies (3.4). Then, with some probability $q[c]$ greater than $(1/8(p(n)))$, the first execution will produce an accepting full transcript with an opening to some value $\sigma \neq \sigma^* = \sigma^*[c]$. The probability that the second execution produces an accepting full transcript with an opening to some $\sigma' \neq \sigma$ is greater $q[c]/2$; otherwise σ would be the most likely opening conditioned on c, contradicting the fact that $\sigma \neq \sigma^*$. Thus, we break the first-phase binding with probability at least $(1/8p(n) \cdot q[c] \cdot q[c]/2 = \Omega(1/p(n)^3)$, contradicting the security of (S, R). □

We the above claims, we complete the proof. By inspection, we have $\Pr[\bigvee_i E_i] = 1$, and thus:

$$\Pr[\mathsf{S}^* \text{ succeeds}] \leq \Pr[E_1] + \sum_{i=1}^{4} \Pr[\mathsf{S}^* \text{ succeeds} \wedge E_i] \leq \frac{1}{2} + \frac{1}{p(n)},$$

as desired. □

Boosting the binding. The commitment scheme $(\mathbb{S}, \mathbb{R})$ from Lemma 3.3.6 is only $(\frac{3}{4} + \mathrm{neg}(n))$-binding. Nonetheless, by the following "folklore" claim, $(\mathbb{S}, \mathbb{R})$ implies a commitment scheme that is $\mathrm{neg}(n)$-binding and preserves the same hiding property as the original scheme.

CLAIM 3.3.13

There exists an efficient procedure that for any function $\delta \geq 1/\mathrm{poly}(n)$ converts a statistically [resp., computationally] $(1 - \delta(n))$-binding commitment scheme $(\mathbb{S}, \mathbb{R})$ into a commitment scheme (S, R) that is statistically [resp., computationally] binding. Furthermore, if $(\mathbb{S}, \mathbb{R})$ is statistically [resp., computationally] hiding, so is (S, R).

Proof. The protocol (S, R) is defined as follows: in order to commit to a bit b, the two parties run $t = \lceil n/\delta \rceil = \mathrm{poly}(n)$ independent executions of the commit stage of $(\mathbb{S}(b), \mathbb{R})$ one after the other, where S and R acting as $\mathbb{S}$ and $\mathbb{R}$ respectively. In the reveal stage, S decommits, via the reveal stage of $(\mathbb{S}, \mathbb{R})$, all the t commitments and R accepts if and only if all the commitments are opened successfully to the same value. The hiding of the above scheme follows by a straightforward hybrid argument. For the binding part, let S^* be a PPT trying to break the binding of (S, R). We show that S^* breaks the

binding of (S, R) only with negligible probability, and since S^* was arbitrarily chosen it follows that (S, R) is computationally binding.

We say that S^* ***breaks the binding of the*** i^{th} ***execution*** of $(\mathbb{S}, \mathbb{R})$ if while trying to break the binding of (S, R) it successfully opens the i^{th} commitment into two different values. Notice that this event depends on several random variables: $C_{<i}$, the coins of S^* and the coins of R in the first $i-1$ executions; C_i, the coins of R in the i'th execution; and $C_{>i}$, the coins of R in executions $i+1, \ldots, t$. For settings $(c_{<i}, c_i) \in \mathrm{Supp}(C_{<i}, C_i)$, we define $q_i(c_{<i}, c_i)$ to be the probability over $C_{>i}$ that S^* breaks the binding of the i^{th} execution conditioned on $(C_{<i}, C_i) = (c_{<i}, c_i)$.

For an arbitrary positive polynomial p, define a prefix $c_{<i}$ to ***bad*** if $\Pr[q_i(c_{<i}, C_i) > 1/p(n)] > 1 - \delta + 1/p(n)$, and otherwise call $c_{<i}$ ***good***. We will now show that $\Pr[C_{<i} \text{ is bad}] \leq 1/p(n)$. Suppose not. Then we can construct an efficient algorithm $\mathbb{S}^*$ that breaks the binding of $(\mathbb{S}, \mathbb{R})$ with probability $1 - \delta + 1/3p(n)$. In the commit stage, $\mathbb{S}^*$ first finds a value $c_{<i}$ for which $\Pr[q_i(c_{<i}, C_i) > 1/2p(n)] > 1 - \delta + 1/2p(n)$ and "hardwires" this value into S^*. (Note that the above can be done efficiently and with overwhelming success probability by random sampling, given oracle access to S^*). When interacting with $\mathbb{R}$, $\mathbb{S}^*$ acts as S^* does in the ith execution of (S^*, R). With probability at least $1 - \delta + 1/2p(n)$ over the coins c_i of $\mathbb{R}$, we have $q_i(c_{<i}, c_i) > 1/2p(n)$. If this occurs, then by randomly continuing the simulation of (S^*, R) with $O(n \cdot p(n))$ independent choices of $C_{>i}$, $\mathbb{S}^*$ will be able to break the binding with probability $1 - \mathrm{neg}(n)$. Thus, $\mathbb{S}^*$ breaks the binding of $(\mathbb{S}, \mathbb{R})$ with probability $1 - \delta + 1/2p(n) - \mathrm{neg}(n) > 1 - \delta + 1/3p(n)$.

Let E_1 be the event that for some i, $C_{<i}$ is bad. By the above and a union bound, $\Pr[E_1] \leq t/p(n)$. Let E_2 be the event that for some i, $q_i(C_{<i}, C_i) \leq 1/p(n)$ but S^* breaks the binding of the i'th execution. By the definition of q_i, we have $\Pr[E_2] \leq t/p(n)$. Finally, we have

$$
\begin{aligned}
&\Pr[S^* \text{ breaks the binding} \wedge \neg E_1 \wedge \neg E_2] \\
&\leq \Pr\left[\bigwedge_{i=1}^{t} [(C_{<i} \text{ good}) \wedge (q_i(C_{<i}, C_i) > 1/p(n))]\right] \\
&= \prod_{i=1}^{t} \Pr\left[(C_{<i} \text{ good}) \wedge (q_i(C_{<i}, C_i) > 1/p(n)) \,\middle|\, \bigwedge_{j<i} [(C_{<j} \text{ good}) \wedge (q_j(C_{<j}, C_j) > 1/p(n))]\right] \\
&\leq (1 - \delta + 1/p(n))^t \\
&= \mathrm{neg}(n) + t/p(n),
\end{aligned}
$$

where the last inequality can be seen by considering any fixed value $C_{<i} = c_{<i}$, which fixes the event on which we are conditioning in the i'th factor and whether $C_{<i}$ is good or bad. If $c_{<i}$ is bad, then the probability in the i'th factor is 0. If $c_{<i}$ is good, then the probability (over just C_i) is at most $(1 - \delta + 1/p(n))$ by the definition of good. Taking $p(n)$ to be an arbitrarily large polynomial, we deduce that S^* breaks the binding with negligible probability. □

Having established the appropriate claims and lemmas, we now state what is achievable from our transformation.

THEOREM 3.3.14
There exist an efficient procedure, call it 2-to-1-FullTransform, that takes as input a security parameter 1^n, a two-phase commitment scheme (S, R) with message lengths $(k_1, k_2) = (n, 1)$, and a family of functions $\mathcal{F} = \bigcup_n \mathcal{F}_n = \{f\colon \{0,1\}^n \to \{0,1\}^{\ell(n)}\}$, and outputs a commitment scheme $(S, R) = \text{2-to-1-FullTransform}((\mathsf{S}, \mathsf{R}), \mathcal{F})$ satisfying the following properties:

- If (S, R) is statistically hiding and $\mathcal{F}$ has the large preimages property, then (S, R) is statistically hiding.
- If (S, R) is statistically [resp., computationally] $\binom{2}{1}$ binding and $\mathcal{F}$ has statistical [resp., computational] target collision resistance, then (S, R) is statistically [resp., computationally] binding (in the standard sense of binding).
- If (S, R) is public coin, then (S, R) is also public coin.

Proof. We describe the 2-to-1-FullTransform algorithm, recapping what we have done thus far, as follows.

1. Apply Algorithm 3.3.1 on (S, R) and $\mathcal{F}$ to obtain a (standard) commitment scheme $(\mathbb{S}, \mathbb{R})$. Lemmas 3.3.3 and 3.3.6 state that for the right properties of both $(\mathsf{S}', \mathsf{R}')$ and $\mathcal{F}$ (see the first two items in 3.3.14 above), $(\mathbb{S}, \mathbb{R})$ is hiding and $(1/2 + \text{neg}(n))$-binding.
2. Next, using Claim 3.3.13, boost the binding of $(\mathbb{S}, \mathbb{R})$ to obtain a scheme (S, R) that is $\text{neg}(n)$-binding while not affecting the hiding property. Output (S, R) as our desired scheme.

As for the preservation of the public coin property, observe that the messages sent by $\mathbb{R}$ that are specific to the 2-to-1-Transform are choosing $f \leftarrow \mathcal{F}$ and selecting $\mathit{phase} \leftarrow \{0,1\}$, both of which are public coin operations. □

3.4 Putting it Together

Now, we put together everything from the previous sections to establish our main theorem.

RESTATEMENT OF THEOREM 3.1.1
Given a one-way function $f : \{0,1\}^n \to \{0,1\}^n$, we can construct in time polynomial in n a public-coin commitment scheme $(\mathbb{S}, \mathbb{R})$ that is statistically hiding and computationally binding.

The statistical hiding property holds regardless of whether or not f is secure (hard to invert). On the other hand, if f is nonuniformly secure, than $(\mathbb{S}, \mathbb{R})$ will be computationally binding with nonuniform security.

Proof of Theorem 3.1.1. We start off by constructing a collection of two-phase commitment schemes from f using Theorem 3.2.13. For any polynomial $k(n)$ (which we will choose below), we can construct in time polynomial in n a collection of $m = \text{poly}(n)$ public-coin two-phase commitment schemes $\mathcal{COM} = \{\mathsf{Com}_1, \cdots, \mathsf{Com}_m\}$ with message lengths $(k(n), 1)$ such that:

- there exists an index $i \in \{1, 2, \ldots, m\}$ such that scheme Com_i is statistically hiding, and

- for every index $i \in \{1, 2, \ldots, m\}$, scheme Com_i is computationally $\binom{2}{1}$ binding.

(As remarked after Theorem 3.2.13, we can obtain two-phase commitments with message lengths $(k(n), k(n))$ for any polynomial k that we choose. Using only 1 bit of the 2nd-phase message (padding with $k-1$ zeroes), we obtain message lengths $(k, 1)$.)

Now in order to apply Theorem 3.3.14, from f we use [Rom90, KK05] to obtain a universal one-way hash family $\mathcal{F}_n = \{f : \{0,1\}^{2k(n)} \to \{0,1\}^{k(n)}\}$ for some polynomial k (which we use to determine the message length for the two-phase commitment above).[1] Let the resulting (standard) commitment schemes be $\mathsf{Com}'_i = \text{2-to-1-}\mathsf{FullTransform}(\mathsf{Com}_i, \mathcal{F})$. By Theorem 3.3.14 and Lemma 3.2.6, we know that:

- Com'_i is statistically hiding if Com_i is statistically hiding,
- Com'_i is computationally binding if Com_i is computationally $\binom{2}{1}$ binding, and
- Com'_i is public coin if Com_i is public coin.

This means that we now have a collection of public-coin (standard) commitment schemes $\mathcal{COM}' = \{\mathsf{Com}'_1, \cdots, \mathsf{Com}'_m\}$, where $m = \text{poly}(n)$, such that:

- there exists an index $i \in \{1, 2, \ldots, m\}$ such that scheme Com'_i is statistically hiding, and
- for every index $i \in \{1, 2, \ldots, m\}$, scheme Com'_i is computationally binding (in the standard sense of binding).

We are almost done, except that we are still left with a collection of commitments instead of a *single* commitment scheme. The following claim states that the latter collection can be converted into the desired commitment scheme.

CLAIM 3.4.1

There is an efficient procedure that converts a polynomial collection of commitment schemes, at least one of which is statistically hiding and all are computationally binding, into a *single* commitment scheme that is statistically hiding and computationally binding. In addition, if we start off with public-coin schemes, we also end up with a public-coin scheme.

Proof. To commit to a bit b, we randomly secret-share $b = b_1 \oplus \cdots \oplus b_m$ and commit to share b_i using the i'th commitment scheme. Alternatively, the proposition can be deduced from [HHK+05, Thm. 5.2]. □

The main theorem statement is now complete since we now have a *single* commitment scheme that is statistically hiding and computationally binding, and the only complexity assumption made is the existence of one-way functions.

We now proceed to the additional properties mentioned. By inspection, we observe that the statistical hiding properties throughout the construction hold regardless of the security of f. As for

[1] Since we are using here the uniform definition of universal one-way hash family (i.e., where x is sampled by A), we need to use the theorem of Katz and Koo [KK05]. In their theorem, however, it is not explicitly defined whether or not the adversary can encode additional information (i.e. state) between the declaration of x and finding the collision (see Remark 3.2.3). Fortunately, the stronger version of this theorem required by our proof, follows readily from their original proof.

nonuniform security, we observe that our construction is "fully black-box" in the sense of [RTV04]; in particular, the computational binding property is proven by specifying for every polynomial p, a PPT reduction R such that if $\mathbb{S}^*$ is *any* sender strategy (of arbitrary complexity) that breaks the binding property with probability with probability $1/p(n)$, then $\mathsf{R}^{\mathbb{S}^*}$ inverts f with non-negligible probability. In particular, if $\mathbb{S}^*$ a nonuniform PPT algorithm, then we obtain a nonuniform PPT inverter for f, which cannot exist if f is non-uniformly secure. □

3.5 Conclusions

While our result resolves the complexity assumption needed to construct statistically hiding commitment schemes, there is still room for substantial improvements. Both the construction and its analysis are rather involved; we hope that a simpler and more direct proof can be found. A related concern is that the construction is very inefficient, and certainly would never be utilized in practice. In particular, given a one-way function $f : \{0,1\}^n \to \{0,1\}^n$, our commitment scheme utilizes $\mathrm{poly}(n)$ invocations of the function f and $\mathrm{poly}(n)$ rounds of interaction. It would be interesting to substantially improve these bounds or argue that they are essentially optimal. It was recently shown in [HHRS07] (generalizing [Sim98, Wee07]) that any black-box construction of statistically hiding commitments from even one-way permutations must have $\Omega(n/\log n)$ rounds of interaction. Our construction, as well as that of Naor et al. [NOVY98], is black box; bypassing the lower bound with a non-black-box construction would be very interesting (even if unlikely to yield something practical).

Chapter 4

A New Interactive Hashing Theorem

4.1 Introduction

Interactive hashing, introduced by Naor, Ostrovsky, Venkatesan and Yung [NOVY98], is a protocol that allows a sender S to commit to a particular value while only revealing to a receiver R some predefined information of this value. More specifically, S commits to a value y while only revealing to R the value $(h, z = h(y))$, where h is some random hash function (we postpone additional details on the choice of hash function). The two security properties of interactive hashing are *binding* (namely, S is bounded by the protocol to at most one value of y) and *hiding* (namely, R does not learn any impermissible information about y). In the following we elaborate more on these security requirements.

Binding. There are several ways in which the above binding requirement can hold. Clearly, binding holds if after the interaction ends there is only a single element y that is consistent with the transcript.[1] In the computational setting, however, we are interested in the case where a random h does induce many collisions. Thus, we only require the binding property to hold against efficient senders. Assuming that h is taken from a family of collision resistant hash functions,[2] designing an interactive hashing protocol with respect to h is straightforward (simply let R send h over to S and in return S can send $h(y)$ to R). The protocols analyzed in this paper will not rely on collision resistant hash functions (as we do not want to assume their existence). Following [NOVY98], we enforce the binding by asking the sender to provide additional information about y (typically, the honest sender gets this additional information as part of its input).

We formally define the computational binding property in Section 4.3, but in the meanwhile let us consider the following important example: let f be a one-way permutation and view the committed value y as an image of f. For the purpose of binding we can now require the sender to provide x such that $y = f(x)$ is consistent with the transcript (i.e., $h(y) = z$, where the transcript equals (h, z)). Thus for breaking the binding of the protocol, a cheating sender needs not only output $y_1 \neq y_2$ such that $h(y_1) = h(y_2) = z$, but it also needs to output x_1 and x_2 for which $f(x_1) = y_1$ and $f(x_2) = y_2$. Indeed with this additional requirement, [NOVY98] constructs

[1] Interactive hashing protocols with this strong binding property (known as "information theoretic" interactive hashing) have many implications including [CCM98, CS06, DHRS04, NV06], to name a few.

[2] $\mathcal{H}$ is a family of collision resistant hash functions if given a random $h \in \mathcal{H}$, it is infeasible to find $x_1 \neq x_2$ for which $h(x_1) = h(x_2)$.

an interactive hashing protocol that allows collisions but is nevertheless binding, more details in Section 4.1.1.

Hiding. Ideally, we would like to have hiding for interactive hashing in the secure-function-evaluation sense. That is, we would like the only information that a receiver acquires about y through the protocol to be its hash value for a uniformly distributed hash function. Forcing such a strong hiding property against malicious receivers, however, requires using stronger machinery that we would like to avoid in this paper. Therefore, we only require such a strong hiding property in the case of *semi-honest* (a.k.a. honest-but-curious) receivers (these are receivers that follow the protocol correctly, but may try to extract additional information from their view of the interaction). In the case of *malicious* receivers, we require that the only information obtained is the hash value of y for some, adversatively and possibly adaptively chosen, hash function.[3]

Perspective. Interactive hashing (in the flavor mentioned above) is closely related and to a large extent motivated by the fundamental notion of statistically hiding (and computationally binding) commitments. Statistically hiding commitments can be used as a building block in constructions of statistical zero-knowledge arguments [BCC88, NOVY98] or certain coin-tossing protocols [Lin03]. Naor et al. [NOVY98] use their interactive hashing protocol, hereafter called the NOVY protocol, in order to construct statistically hiding commitments based on any one-way permutation. Haitner et al. [HHK+05] generalized their result by showing that the NOVY protocol can be used to construct statistically hiding commitments based on regular one-way functions and also on the so called approximable-size one-way functions. Independently of this work, Haitner and Reingold [HR07], using the "one out of two binding" commitment scheme of Nguyen et al. [NOV06], constructed statistically hiding commitments based on *any* one-way functions. Not surprisingly, interactive hashing is heavily used in the underlying commitment scheme of [NOV06]. Interactive hashing is also used by several other cryptographic protocols [GGL98, OVY93a, OVY92, OVY93b].

A possible drawback of [NOV06], and thus of [HR07], is that their construction is rather inefficient and complex. Indeed, a major motivation for looking into interactive hashing is to simplify constructions of statistically hiding commitment schemes based on any one-way functions. Before discussing our results and their applications, let us have a closer look into the notion of interactive hashing.

4.1.1 Interactive Hashing in the Setting of One-Way Permutations

Let $f\colon \{0,1\}^n \mapsto \{0,1\}^n$ be a one-way permutation and consider the following two-party protocol between a sender S, getting as input $x \in \{0,1\}^n$ and $y = f(x)$, and a receiver R: the receiver selects a random two-to-one hash function $h\colon \{0,1\}^n \mapsto \{0,1\}^{n-1}$ and sends its description to S, and S sends $z = h(y)$ back to R. Note that if both parties follow the protocol, then the following binding property is guaranteed: It is not feasible for S to find $x' \in \{0,1\}^n$ such that $f(x') \neq f(x)$ but $h(f(x')) = h(f(x)) = z$, although (exactly one) such element x' does exist. The reason is that the task of finding such x' can be shown to be equivalent in hardness to inverting f on a random image (whereas the latter task is assumed to be hard by the one-wayness of f). Furthermore,

[3] We mention that in various settings assuming the existence of one-way functions one can assume without loss of generality that the receiver is semi-honest (see [GMW91]). In particular, this is the case for statistically hiding commitments ([HHK+05, Theorem 6.1]).

we are guaranteed to have the following hiding property: let y_1 and y_2 be the two preimages of z with respect to h. Given R's view of the communication (i.e., given the values h and z), it is indistinguishable whether the second input element of S (i.e., y) is y_1 or y_2.

What happens, however, if S selects x only *after* seeing h? In such a case, it is quite plausible that S would be able to "cheat" by producing $x, x' \in \{0,1\}^n$ such that $f(x) \neq f(x')$ but $h(f(x')) = h(f(x)) = z$.[4] The NOVY interactive hashing protocol prevents such cheating. For that it employs a specific family of hash functions such that each one of its functions h can be decomposed into $n-1$ Boolean functions $h_1, \ldots, h_{n-1}$, where $h(x) = h_1(x), \ldots, h_{n-1}(x)$.[5] In the NOVY protocol, instead of sending h at once as described above, the protocol proceeds in rounds such that R sends a single Boolean function h_i in each round, and in return the sender sends a bit z_i, which is supposed to equal $h_i(f(x))$. Intuitively, a cheating sender has a significantly smaller leeway for cheating as it can no longer wait in selecting x till it receives the entire description of h. Still, it is highly non-trivial to argue that restricting the sender by adding interaction as above is sufficient in order to prevent the sender from cheating. Nevertheless, Naor et al. [NOVY98] have shown that their protocol has the binding property even against a cheating sender (namely, even a cheating sender cannot produce $x, x' \in \{0,1\}^n$ such that $f(x) \neq f(x')$, but $h(f(x')) = h(f(x)) = z$).

REMARK 4.1.1 (application to statistically hiding commitments)
Assuming that one-way permutations exist, the NOVY protocol applies the existence of statistically hiding commitments as follows:[6] we employ the NOVY protocol with a uniformly chosen pair $(x, y = f(x))$ as the sender's input. Let y_0 and y_1 be the two preimages of the hash value determined by the protocol (i.e., $y_0 \neq y_1 \in h^{-1}(z)$) and let $i \in \{0,1\}$ be such that $y = y_i$. The sender commits to a bit $b \in \{0,1\}$ by masking it with i (i.e., it sends $c = i \oplus b$ to the receiver). In order to decommit, the sender sends (x, y) to the receiver, who sets $b = c \oplus i$, where i is as above (e.g., the index of y in $\{y_0, y_1\}$). It is easy to verify that the binding and hiding this commitment scheme follow by the security of the NOVY protocol discussed above.

4.1.2 Interactive Hashing in the Sparse Case

How about constructing statistically hiding commitments from, say, regular one-way functions (one-way functions where every possible output has the same number of preimages)? In such a case we would like to interactively hash a value y (an output of the one-way function) that is taken from some subset $L(= Im(f))$ of $\{0,1\}^n$ (and not from $\{0,1\}^n$ as in the case of one-way permutations). Notice that the NOVY theorem guarantees that when hashing y with $h\colon \{0,1\}^n \mapsto \{0,1\}^{n-1}$ the sender is committed to a single value y. In the case of sparse L, however, when h outputs so many bits then it is most likely that $h(y)$ completely determines y, and we cannot employ the NOVY protocol (at least not as we did in Remark 4.1.1), to get statistically hiding commitments.

Facing the aforementioned difficulty, Haitner et al. [HHK+05] make the following observation: the binding of the NOVY protocol holds for every function f that it is hard to invert over the uniform

[4] Assume for example that the one-way permutation equals the identity function on the set T of all strings that start with $n/4$ zeros (where n is the input length). Now given a hash function h all that the cheating sender has to do is to find a collision $y_1 \neq y_2$, where $y_1, y_2 \in T$, such that $h(y_1) = h(y_2)$. Such a collision is likely to exist by the birthday paradox, and for many families of hash functions (e.g., linear transformations) finding such a collision is easy.

[5] For more details on the definition of this family of hash functions see Section 4.6.

[6] Actually, the NOVY commitment schemes are even stronger being perfectly hiding.

distribution on $Im(f)$ (in particular, one-way permutations have this property), where some weak hiding is guaranteed for every f such that $Im(f)$ is "dense" in $\{0,1\}^n$ (i.e., of order $2^n/\operatorname{poly}(n)$). Equipped with this observation, [HHK+05] employ the NOVY protocol with length-preserving poly-to-one one-way function (i.e., each output has at most polynomial number of preimages in the image set of f), to get some weak form of statistically hiding commitments that can later be amplified to full-fledge statistically hiding commitments. To handle any regular one-way function, [HHK+05] applies additional layer of (non interactive) hashing to reduce to the dense case. This implies a construction of statistically hiding commitments from any regular one-way function with known image size. Interactive hashing in the sparse case arises in other works as well, most notably in the construction of statistical zero-knowledge arguments from any one-way function [NOV06].

4.1.3 Our Results

We introduce an alternative proof for the NOVY protocol that relies in parts on the original proof due to [NOVY98] (the NOVY proof), but still seems to us significantly simpler. The proof follows a simple intuition that is sketched in Section 4.1.6. Moreover, the parameters achieved by our proof are an improvement compared with the original ones. Given an algorithm A that breaks the binding property with probability $\varepsilon_A(n)$, we get an algorithm that inverts the one-way permutation in comparable time and with inverting probability $\frac{\varepsilon_A^2(n)}{\operatorname{poly}(n)}$ (where n is the hash function input length). This is a substantial improvement over the $\varepsilon_A^{10}(n)\cdot\operatorname{poly}(n)$ reduction of [NOVY98] and is much closer to natural limitations of the proof technique (see discussion in Section 4.7).[7]

In addition to being simpler and more security preserving, the new proof implies a more general interactive hashing theorem. The new theorem applies to every family of hash functions that is a "product" of Boolean families of pairwise independent hash functions (i.e., $f(x)=(h_1(x),\ldots h_k(x))$ where $h_1\ldots h_k$ are taken from Boolean families of pairwise independent hash functions) and not only to the special family of two-to-one hash functions used by [NOVY98, NOV06]. More importantly, the new theorem directly applies to the "sparse case". Let $f\colon\{0,1\}^n\mapsto\{0,1\}^n$ be an efficiently computable function and let $L\subseteq\{0,1\}^n$ be sparse (i.e., $|L|/2^n=\operatorname{neg}(n)$). As mentioned above, when hashing a value $y\in L$, the NOVY proof only promises binding when using a hash function that outputs almost n bits. In such a case, however, y is likely to be completely determined by $h(y)$ (which makes the protocol not useful for these settings). Our theorem applies as long as hashing is to roughly $m=\lfloor\log(|L|)\rfloor$ bits. In particular, when h is taken from a family of hash functions $\mathcal{H}^m\colon\{0,1\}^n\mapsto\{0,1\}^m$ that is a product of m families of pairwise independent Boolean hash functions, we can show that a close variant of the NOVY protocol possesses the following binding property: if f is hard to invert on the uniform distribution over L, then no polynomially-bounded sender S^* (even one which arbitrarily deviates from the protocol) can find two elements $x,x'\in f^{-1}(L)$ such that $f(x)\neq f(x')$ but $h(f(x))=h(f(x'))=z$ (where z is the value determined by the protocol as $h(y)$).

Finally, we consider interactive hashing protocols that use pairwise independent hash functions of arbitrary output length (and not necessarily Boolean). Given a one-way function that is hard to invert over the uniform distribution by algorithms of running-time $2^{s(n)}$, our approach yields an interactive hashing protocol with $O(n/s(n))$ rounds. When applied to one-way permutations, the resulting protocol matches the recent lower bound of Haitner et al. [HHRS07],[8] and generalizes the

[7] We mention that independently of our work, [NOV06] presented an $\frac{\varepsilon_A^3(n)}{\operatorname{poly}(n)}$ reduction.

[8] [HHRS07] show that in every fully-black-box reduction from interactive hashing protocol to $s(n)$-hard permuta-

one-way permutations based $O(n/\log(n))$-rounds protocol recently given by [KS06].

As an easy corollary, we use the new theorem to derive a direct construction of statistically hiding commitment based on regular one-way functions of known regularity (and thus reprove [HHK+05]). We also believe that our new result can be used to simply the construction of two-phase commitment schemes given in [HNO+07], and thus to simplify their construction of statistically hiding commitments from any one-way functions.

4.1.4 Related Work

We mention that independently of our work, Nguyen et al. [NOV06] gave a new proof for the NOVY protocol. Their proof follows the proof of [NOVY98] more closely than ours but still introduces various simplifications and parameter improvements. The main goal of their new proof was to generalize the protocol such that it allows hashing with a hash function that is poly-to-one rather than two-to-one as in [NOVY98]. In other words, they analyze the NOVY protocol with $n-O(\log n)$ rounds rather than $n-1$ rounds in [NOVY98].[9] For a comparison between the parameters obtained by [NOVY98], [NOV06] and our work, see Remark 4.4.6.

4.1.5 Generalization to Relations

Following [NOV06], we state our protocol, and analysis, in the more general setting of binary relations rather than functions. That is, we consider a binary relation W that is hard to satisfy (i.e., given y it is hard to find x such that $(x,y) \in W$).[10] For such a relation we prove that following our interactive hashing protocol, S cannot find two pairs $(x_0,y_0),(x_1,y_1) \in W$ such that both y_0 and y_1 are consistent with the protocol, but $y_0 \neq y_1$. We mention that since every function, f, defines the natural binary relation $\{(x \in \{0,1\}^*, f(x))\}$, every result with respect to binary relations implies an equivalent result with respect to functions. Finally, we stress that the relations we consider might not be efficiently verifiable - deciding membership in the given relation might be hard.

4.1.6 Proof Idea

We outline our binding proof in the most basic setting where $f\colon \{0,1\}^n \mapsto \{0,1\}^n$ is a one-way permutation and $L = \{0,1\}^n$ (the proof of general case is not significantly different). Let $x \in \{0,1\}^n$ be S's input and let $y = f(x)$. Our protocol consists of $m = n - O(\log(n))$ rounds, in each round R selects a random Boolean pairwise-independent hash function h_i and S replies with $z_i = h_i(y)$. Let A be an algorithm that plays the sender's role in the protocol and at the end of the protocol outputs two elements $x_1, x_2 \in \{0,1\}^n$. Assume that with some noticeable probability ε, it holds that $f(x_1) \neq f(x_2) \in \mathsf{Consist}(h_1,\ldots,h_m)$, where $\mathsf{Consist}(h_1,\ldots,h_m)$ is the set of elements that are consistent with the transcript of the protocol (i.e., $\mathsf{Consist}(h_1,\ldots,h_m) \stackrel{\text{def}}{=} \{y' \in \{0,1\}^n \colon \forall i \in [m]\ h_i(y') = z_i\}$). Consider the following naive way one may try to invert f using A: given input y, the algorithm

tions (hard for algorithms of running-time at most $2^{s(n)}$), the protocol has $\Omega(n/s(n))$ rounds.

[9] We mention that such a generalization also follows by our analysis, which considers any value for m - the protocol's number of rounds.

[10] Since we are generalizing relations naturally defined by one-way functions (i.e., $W = \{(x,f(x))\}$), we chose to write the witness (i.e., x) as the left hand side parameter, not following the more traditional notation of putting the witness on the right hand side.

chooses the hash functions at random and returns one of the two values that A outputs. It is rather easy to argue that this naive algorithm inverts f with probability at least $\frac{\varepsilon}{2^n}$.[11]

Assume for a second that we are only trying to invert f on the following distribution: the first $k = m - \log(\frac{1}{\varepsilon}) - c \cdot \log(n)$ (for some constant $c \in \mathbb{N}$ determined by the analysis) Boolean hash functions, $h_1, \ldots, h_k$, are chosen at random and only then a random element, y, is uniformly drawn from $\mathsf{Consist}(h_1, \ldots, h_k)$. We call the distribution induced on $(y, h_1, \ldots, h_k)$ by the above process D_{Uni}. On the average, A has probability ε to cheat even when conditioned on $h_1, \ldots, h_k$ being selected. Also note that by the pairwise independence of the h's, the size of $\mathsf{Consist}(h_1, \ldots, h_k)$ is, with high enough probability, about $\frac{2^n}{2^{m-k}} = \frac{n^c}{\varepsilon}$. It follows that the naive algorithm, which selects the rest of the hash functions at random and returns one of A's answers, inverts f with probability close to $\frac{\varepsilon^2}{n^c}$ over D_{Uni}.

Now let's try to emulate the above setting on a random $y \in \{0,1\}^n$. Namely, given a uniformly chosen $y \in \{0,1\}^n$, we will choose $h_1, \ldots, h_k$ so that $(y, h_1, \ldots, h_k)$ will have about the same distribution as it was drawn from D_{Uni}. To do so, we choose $h_1, \ldots, h_k$ one by one, each time we keep sampling until we find an hash function that its value on y is consistent with A's answer (if the answer is inconsistent, we "rewind" A to its state before it was asked the last "faulty" hash function). We call D_{Sim} the distribution the above process induces on $(y, h_1, \ldots, h_k)$. Assuming that we could prove that the statistical difference between D_{Uni} and D_{Sim} is smaller than $\frac{\varepsilon^2}{n^c}$ (recall that this is the inverting probability of the naive algorithm on D_{Uni}), we could easily conclude the proof of the binding property. Unfortunately, we cannot prove such a strong bound.

Note that till now we did not take advantage of the full power of A, since we did not use the fact that A finds two different outputs of f that are consistent with the protocol and not merely a single one (indeed the above observations hold also with respect to the honest S). When taking into account the full powers of A, we manage to prove that the success probability of the naive algorithm with respect to D_{Uni} does not depend on inverting too few elements. More specifically, the subset of $y \in \mathsf{Consist}(h_1, \ldots, h_k)$ such that the naive algorithm inverts on them with "high enough" probability is of relative size $\sqrt{\varepsilon \cdot |\mathsf{Consist}(h_1, \ldots, h_k)|}$.[12]

The latter observation turns to be useful, since we also mange to prove the following: for most choices of $h_1, \ldots, h_k$ (excluding a set of probability much smaller than $\frac{\varepsilon^2}{n^c}$), and for most elements in $\mathsf{Consist}(h_1, \ldots, h_k)$ (excluding a set of size much smaller than $\sqrt{\varepsilon \cdot |\mathsf{Consist}(h_1, \ldots, h_k)|}$), the probability mass that $(y, h_1, \ldots, h_k)$ has under D_{Uni} is within a constant factor from its mass under D_{Sim}. By the above observations, it follows that we can invert y with noticeable probability over D_{Sim}, which directly implies that we can invert f (again, with noticeable probability) on the uniform distribution over $\{0,1\}^n$.

[11] With probability $|\mathsf{Consist}(h_1, \ldots, h_m)| / 2^n$, a random y lands in $\mathsf{Consist}(h_1, \ldots, h_m)$ (and is uniform there). Now if $f(x_1)$ or $f(x_2)$ are in $\mathsf{Consist}(h_1, \ldots, h_m)$ (which clearly happens with probability at least ε), then with probability at least $1/|\mathsf{Consist}(h_1, \ldots, h_m)|$ either x_1 or x_2 are the inverse of y. All in all, we invert y with probability $\frac{|\mathsf{Consist}(\mathcal{H}_1, \ldots, h_m)|}{2^n} \cdot \varepsilon \cdot \frac{1}{|\mathsf{Consist}(h_1, \ldots, h_m)|} = \frac{\varepsilon}{2^n}$.

[12] Loosely, let T be the set of y's that A is likely to output their inverse (according to f). A random selection of $h_{k+1}, \ldots, h_m$ separates every two elements in T with probability $1 - 2^{-(m-k)}$. So unless the size of T is large enough, one of the two values A output will be forced to be the inverse of an element outside of T. This will contradict the assumptions that values outside of T are only inverted with small probability.

4.1.7 Outline

We give our definition of interactive hashing in Section 4.3 and present an interactive hashing protocol that satisfies this definition in Section 4.4. In Section 4.5 we generalize the result of Section 4.4 for the case of non-Boolean hash functions, where in Section 4.6 we argue that the new proof can also be applied to the original NOVY protocol (that uses very specific hash functions). Discussion and further issues appear in Section 4.7. Finally, in Section 4.8 we show how to use our new theorem to derive a direct construction of statistically hiding commitment based on known regular one-way functions.

4.2 Preliminaries

4.2.1 Hard to Satisfy Relations

For a binary relation, W and a string $y \in \{0,1\}^*$, we denote the set $\{x \in \{0,1\}^* : W(x,y) = 1\}$ by W_y. The following definition is a natural generalization of the hardness imposed by the hardness of a one-way permutation f on the relation $\{(x, f(x))\}$.

DEFINITION 4.2.1 (hard to satisfy relations)
Let W be a binary relation, let $L \subseteq \{0,1\}^n$ and let $s\colon \mathbb{N} \mapsto \mathbb{N}$. We say that W is s-hard to satisfy over L if for every algorithm of running-time at most $s(n)$ it holds that $\Pr_{y \leftarrow L}[A(y) \in W_y] < \frac{1}{s(n)}$. We say that W is hard to satisfy over L (omitting the parameter s), if it is p-hard to satisfy over L for every polynomial p.

4.3 Interactive Hashing - Definition

We choose (following [NOV06]) to state our definitions in the setting of binary relations. This generalizes the original definition due to [NOVY98], which concentrates on the particular relations that are naturally defined by one-way permutations. In particular, the underlying relation is not necessarily efficiently computable or even not efficiently verifiable. Moreover, the relation is not necessarily defined over all strings of a given length, but might rather be defined over some small subset of the strings.

DEFINITION 4.3.1 (interactive hashing protocol)
Let $\mathcal{H}$ be a family of hash functions mapping strings of length n to strings of length $\ell(n)$. An $\mathcal{H}$-interactive hashing protocol $\mathsf{IH} = (\mathsf{S}, \mathsf{R})$ is a probabilistic polynomial-time interactive protocol. Both parties receive the security parameter 1^n as an input and S gets as a private input $y \in \{0,1\}^n$. At the end, S locally outputs y and R outputs $(h, z) \in \mathcal{H} \times \{0,1\}^{\ell(n)}$. We require that for every value of $y \in \{0,1\}^n$ the value of h, induced by the random-coins of S and R, is uniformly distributed in $\mathcal{H}$ and make the following correctness requirement: for all n, all $y \in \{0,1\}^n$, and every pair $(y,(h,z))$ that may be output by $(\mathsf{S}(1^n, y), \mathsf{R}(1^n))$, it is the case that $h(y) = z$.

The security of interactive hashing protocol has two aspects. Binding the sender to y and concealing some information regarding y from R. In this work we focus on security with respect to

polynomially-bounded sender and unbounded receiver. The setting where both the receiver and the sender are unbounded, called *information theoretic interactive hashing* (a.k.a. *interactive hashing for static sets*), is not treated by this work (for details on the information theoretic setting see for example [CCM98, CS06, DHRS04]). We start by formalizing the hiding property.

4.3.1 Hiding

We use two incomparable hiding definitions. For the semi-honest receiver we require (Definition 4.3.2) that the only information it acquires through the protocol about y is its hash value for a uniformly chosen hash function. For malicious receivers, we require (Definition 4.3.3) that the only information obtained is the hash value of y for some, possibly adaptively chosen, hash function.

DEFINITION 4.3.2 (hiding against semi-honest receivers)
Let $\mathcal{H}$ be an efficient family of functions defined over strings of length n and let $\mathsf{IH} = (\mathsf{S}, \mathsf{R})$ be an $\mathcal{H}$-interactive-hashing protocol. We say that IH is hiding against semi-honest receivers, if there exists a polynomial-time simulator Sim such that for every $y \in \{0,1\}^n$ the distributions $\mathrm{view}^{\pi}_{\mathsf{R}}(\mathsf{S}(y), \mathsf{R})(1^n)$ and $\mathsf{Sim}(1^n, h, h(y))_{h \leftarrow \mathcal{H}}$ are identical.

DEFINITION 4.3.3 (weakly hiding)
Let $\mathcal{H}$ be an efficient family of functions defined over strings of length n and let $\mathsf{IH} = (\mathsf{S}, \mathsf{R})$ be an $\mathcal{H}$-interactive-hashing protocol. We say that IH is weakly hiding if for every algorithm R^* there exist two polynomial-time algorithms Ext and Sim such that the following hold:

1. given an interaction of R^* with $\mathsf{S}(y)$, algorithm Ext outputs $h \in \mathcal{H}$.
2. for $r \in \{0,1\}^*$ let R^*_r denote the deterministic algorithm obtained from R^* by fixing its random-coins to r, and let $\mathrm{Ext}(r, y)$ denote the output of Ext that follows the interaction of R^*_r with $\mathsf{S}(y)$ (note that $Ext(r, y)$ is a random-variable that depends on the random-coins Ext and S). Then for every possible value of $r \in \{0,1\}^*$ and $y \in \{0,1\}^n$, it holds that $\mathrm{view}^{\pi}_{\mathsf{R}^*}(\mathsf{S}(y), \mathsf{R}^*_r)(1^n)$ and $\mathsf{Sim}^{\mathsf{R}^*_r}(1^n, Ext(r, y))$ are identical.

4.3.2 Binding

DEFINITION 4.3.4 (binding)
Let $L \subseteq \{0,1\}^n$, let W be a binary relation, let $\mathcal{H}$ be a family of hash functions mapping strings of length n to strings of length $\ell(n)$ and finally let $\mathsf{IH} = (\mathsf{S}, \mathsf{R})$ be an $\mathcal{H}$-interactive-hashing protocol. We say that IH is (computationally) binding for L and W, if no PPT S^* succeed in the following game with more than negligible probability.

> On security parameter 1^n, S^* interacts with R and R outputs (h, z). Then S^* outputs pairs $(x_0, y_0), (x_1, y_1) \in W$ such that $y_0 \neq y_1 \in L$ and $h(y_0) = h(y_1) = z$.

4.4 The New Theorem

In this section we give a construction and new proof that match the definition of Section 4.3.

THEOREM 4.4.1
Let $L \subseteq \{0,1\}^n$, let W be a binary relation that is hard to satisfy over L and let $m(n) \geq \log(|L|) - O(\log(n))$. Then for some efficient family $\mathcal{H}$ of functions from $\{0,1\}^n$ to $\{0,1\}^{m(n)}$, there exists an $\mathcal{H}$-interactive hashing protocol that is binding with respect to with respect to L and W, hiding against semi-honest receivers and weakly-hiding.

Proof. Consider the following interactive hashing protocol.

4.4.1 The interactive hashing protocol

Let $m\colon \mathbb{N} \mapsto \mathbb{N}$ and let $\mathcal{H}$ be a family of functions mapping strings of length n to strings of length $\ell(n)$.

PROTOCOL 4.4.2
Interactive-hashing protocol $\mathsf{IH} = (\mathsf{S}, \mathsf{R})$.

Common input: 1^n.

S's inputs: $y \in \{0,1\}^n$.

1. For $i = 1$ to $m(n)$:
   (a) R chooses uniformly at random $h_i \in \mathcal{H}$ and sends its description to S.
   (b) If $h \notin \mathcal{H}$, S aborts,
   otherwise, S sends $z_i = h_i(y)$ back to R.
2. S locally outputs y.
3. R outputs $(\overline{h}, \overline{z}) = ((h_1, \ldots, h_{m(n)}), (z_1, \ldots, z_{m(n)}))$.

It it straight forward that IH is an $\mathcal{H}^{m(n)}$-interactive-hashing protocol for every efficient family $\mathcal{H}$. In Section 4.4.2 (Lemma 4.4.3), we show that IH is hiding for every efficient family $\mathcal{H}$. Finally, in Section 4.4.2 (Corollary 4.4.7), we complete the proof by showing that IH is binding with respect to L and W, assuming that $\mathcal{H}$ is a Boolean efficient family of pairwise independent hash functions and $m(n) \geq \log(|L|) - O(\log(n))$. □

4.4.2 Hiding

LEMMA 4.4.3
For every efficient family $\mathcal{H}$, Protocol 4.4.2 is hiding against semi-honest receivers, and weakly hiding.

Proof. The fact that Protocol 4.4.2 is hiding against semi-honest receivers is immediate (Sim on input $(1^n, h, h(y))$ outputs $(h, h(y))$), to prove that it is also weakly hiding (against arbitrary adversaries) we define algorithm Ext and Sim as follows: given an interaction of an adversarial receiver R^* with S, let $h_1, \ldots, h_k$ be the questions of R^* answered by S (i.e., $k = m$ if S does not abort). Algorithm Ext selects arbitrary functions $h_{k+1} \ldots, h_m \in h$ and outputs $\overline{h} = (h_1 \ldots, h_m)$.

Given $(h_1 \ldots, h_m)(y)$, algorithm Sim simulates the interaction of R_r^* with $\mathsf{S}(y)$ as follows: let h_i' be the i'th question of R_r^*, if $h_i' \notin \mathcal{H}$ then Sim aborts the execution and output the current view of R_r. Otherwise, Sim forwards $h_i(y)$ to R_r^* as S's answer. Finally, assuming the interaction was not previously aborted, Sim outputs the current view of R_r. It is easy to verify that the view generated by the Sim is identical to the view of R_r^* in the real interaction with $\mathsf{S}(y)$. □

4.4.3 The Main Lemma - Binding

LEMMA 4.4.4

Let $L \subseteq \{0,1\}^n$, W be a binary relation, $m(n) \leq \lfloor \log(|L|) \rfloor$ and $\mathcal{H}$ be an efficient family of pairwise independent Boolean hash functions defined over strings of length n. Finally, let IH be the instantiation of Protocol 4.4.2 with the above $\mathcal{H}$ and m, and let A be an algorithm that runs in time $t_A(n)$ and breaks the binding of IH with respect to W and L with probability $\varepsilon_A(n)$.

Then there exists an oracle algorithm $M^{(\cdot)}$ that given an oracle access to A, the following holds for large enough n.

$$\Pr_{y \leftarrow L}[M^A(y) \in W_y] \in \Omega(\tfrac{2^{m(n)}}{|L|} \cdot \tfrac{\varepsilon_A(n)^2}{n^8}) \ .$$

Letting $t_{\mathcal{H}}(n)$ be an upper bound of the sampling and computing time of $\mathcal{H}$, the running-time of M^A is $O(\log(n)(t_A(n) + m(n)t_{\mathcal{H}}(n)))$.

REMARK 4.4.5

The constrain in the above lemma that $m(n) \leq \lfloor \log(|L|) \rfloor$ was only done for keeping the expressions simple. Using a simple simulation argument, the binding for $m(n) > \lfloor \log(|L|) \rfloor$ can be reduced to the case of $m \leq \lfloor \log(|L|) \rfloor$. We also point out that M^A does not need to know L, W or ε_A.

REMARK 4.4.6 (comparing to [NOVY98] and [NOV06])

The binding proofs [NOVY98] and of [NOV06] hold only in the settings where $L = \{0,1\}^n$, W is the relation naturally defined by a one-way permutation, $\mathcal{H}$ is a specific type family of two-to-one Boolean hash functions and $m(n) = n - 1$ [13]. In the following we compare the success probability and running-time of the inverters the different proofs yield with respect to the above settings.

Let A be an adversary that breaks the binding of the protocol with probability ε_A. Assuming that we use hash functions with linear sampling and evaluation time, the success probability and running-time of our inverter are $\Omega(\frac{\varepsilon_A(n)^2}{n^8})$ and $O(\log(n)t_A(n) + m(n)\log(n)n)$ respectively. The same holds also when the "NOVY hash functions" are used and not Boolean pairwise independent hash function (see Section 4.6). This is an improvement in parameters compared with the analysis in [NOV06, Lemma B.2]. There the algorithm runs in time $O(nT_A(n) + mn^2)$ and breaks f with probability $\Omega(\frac{\varepsilon_A(n)^3}{n^6})$. Finally, in [NOVY98, Lemma 2] the algorithm runs in time $O(nT_A(n)+mn^2)$ (same as in [NOV06]) and only guarantees to break f with probability $\Omega(\frac{\varepsilon_A(n)^{10}}{n^8})$.

In the case of efficient adversaries, Lemma 4.4.4 yields the following corollary.

[13] Actually, the proof of [NOV06] also holds with respect to $m(n) = n - O(\log n)$

COROLLARY 4.4.7
Let L, W, $m(n)$ and IH be as in Lemma 4.4.4. Assuming that W is hard to satisfy over L and that $m(n) \geq \log(|L|) - O(\log(n))$, then IH is binding with respect to L and W.

Proof. We assume without lost of generality that $m \leq \lfloor \log(|L|) \rfloor$ (see Remark 4.4.5) and let A be an efficient algorithm that breaks the binding of IH with non-negligible probability ε_A. Lemma 4.4.4 yields the existence of an efficient algorithm M^A such that $\Pr_{y \leftarrow L}[M^A(y) \in W_y] > \frac{1}{2^{O(\log(n))}} \cdot \frac{\varepsilon_A(n)^2}{n^8} = \frac{\varepsilon_A(n)^2}{\text{poly}(n)}$, contradicting the hardness of W. □

Proof. (of Lemma 4.4.4) We start by arguing that it suffices to consider only deterministic A. Given a randomized algorithm A that breaks the binding of IH when instantiated with a family of hash functions $\mathcal{H}$, consider the family of hash function $\mathcal{H}'$ obtained from $\mathcal{H}$ by appending random strings of length $t_A(n)$ to the hash functions' description. Clearly, the deterministic algorithm A' that emulates A using the randomness given in the hash functions description, breaks the binding of IH instantiated with $\mathcal{H}'$, with exactly the same probability as A does with respect to $\mathcal{H}$. In the following proof we assume nothing about $\mathcal{H}$ but being pairwise independent. Thus, the assumption that A is deterministic is indeed without lost of generality.

For simplicity we drop the dependency on n whenever it is clear from the context. We use throughout the proof the following random variables: for $k \in [m]$ and $\overline{h} \in \mathcal{H}^k$, let $A^{Com}(\overline{h}) \in \{0,1\}^k$ be A's answers when questioned by $\overline{h}$ and let $\mathsf{Consist}(\overline{h}) = \left\{y \in L \,:\, \overline{h}(y) = A^{Com}(\overline{h})\right\}$ (i.e., the set of y's that are consistent with A's answers). Finally, we assume without loss of generality that following any sequence of questions $\overline{h} \in \mathcal{H}^m$, A outputs two pairs of elements $(x_0, y_0), (x_1, y_1) \in \{0,1\}^* \times \{0,1\}^n$ and denote them by $A^{Dec}(\overline{h})$. For $\text{ofs} = \max\{m, \lceil 8\log(n) + \log(1/\varepsilon_A)\rceil + 13\}$, we consider the following algorithm for satisfying W on L.

ALGORITHM 4.4.8
Algorithm M^A.

Input: $y \in L$

1. Let $\overline{h} \leftarrow \mathsf{Searcher}(y)$.
2. Return $\mathsf{Inverter}(\overline{h})$.

Algorithms Searcher and Inverter are defined as follows.

ALGORITHM 4.4.9
Algorithm Searcher.

Inputs: $y \in L$.

1. For $k = 1$ to $m - \text{ofs}$:
   Do the following $2\log(n)$ times:
   (a) Set a value for h_k uniformly at random in $\mathcal{H}$.
   (b) If $A^{Com}(h_1, \ldots, h_k)_k = h_k(y)$, break the inner loop.

2. Return $(h_1, \ldots, h_{m-\text{ofs}})$.

ALGORITHM 4.4.10
Algorithm Inverter.

Inputs: $\overline{h} \in \mathcal{H}^{m-\text{ofs}}$.

1. Choose uniformly at random $\overline{h^e} \in \mathcal{H}^{\text{ofs}}$.
2. Set $((x_0, y_0), (x_1, y_1)) \leftarrow A^{Dec}(\overline{h}, \overline{h^e})$.
3. Return x_0 with probability half and x_1 otherwise.

REMARK 4.4.11
The value ofs depends in our proof on ε_A. This seems to contradict Remark 4.4.5 that M^A does not need to know ε_A. Nevertheless, ofs can instead be selected at random with only a factor m decrease in the success probability of M^A. More interestingly, setting ofs $= 0$ will also guarantee M^A the succuss probability claimed in the theorem. The only affect of decreasing ofs to zero is that $\overline{h^e}$ will be selected by the rewinding method of Searcher rather than uniformly at random by Inverter. For every value $\overline{h^e}$ that satisfies $y \in \mathsf{Consist}(\overline{h}, \overline{h^e})$, we have that the probability of selecting it with the rewinding technique is only larger than the probability of uniformly selecting it. A value of $\overline{h^e}$ such that $y \notin \mathsf{Consist}(\overline{h}, \overline{h^e})$ will not contribute in our analysis to the success probability of M^A. It follows that the distinction between Searcher and Inverter is not necessary for the proof. Still, following [NOVY98], we find this distinction very useful for pedagogical reasons.

Assuming that we use the proper data structure to support the rewinding action, it follows that the running time of M^A is $O(\log(n)t_A(n) + m\log(n) \cdot t_{\mathcal{H}}(n))$. We assume without loss of generality that $m \geq \lceil 8\log(n) + \log(1/\varepsilon_A) \rceil + 13$, otherwise we can set ofs $= m$ and conclude the proof of the theorem directly as follows.

$$\begin{aligned}
&\Pr_{y \leftarrow L}[M^A(y) \in W_y] = \sum_{y \in L} \frac{1}{|L|} \cdot \Pr[\mathsf{Inverter}(H^m) \in W_y] \\
\geq\ & \frac{1}{|L|} \sum_{y \in L} \frac{1}{2} \cdot \Pr\left[((x_0, y_0), (x_1, y_1)) \leftarrow A^{Dec}(H^m)\ :\ x_0 \in W_y \vee x_1 \in W_y\right] \\
\geq\ & \frac{\varepsilon_A}{|L|} \in \Omega\left(\frac{2^m}{|L|} \cdot \frac{\varepsilon_A^2}{n^8}\right)
\end{aligned}$$

We consider the success probability of A with respect to the following distributions.

DEFINITION 4.4.12

- $D_{\text{Sim}} \stackrel{\text{def}}{=} (\overline{h}, y)_{y \leftarrow L, \overline{h} \leftarrow \mathsf{Searcher}(y)}$

- $D_{\text{Uni}} \stackrel{\text{def}}{=} (\overline{h}, y)_{\overline{h} \leftarrow \mathcal{H}^{m-\text{ofs}}, y \leftarrow \mathsf{Consist}(\overline{h})}$

Given that y is uniformly chosen in L, then D_{Sim} is the distribution that Inverter is invoked upon through the execution of M^A. Thus, the probability that Inverter satisfies W over D_{Sim} equals to the success probability of M^A. On the other hand, it is rather easy to show that the probability that Inverter satisfies W over D_{Uni} is noticeable (as a function of ε_A). Intuitively, this is because the distribution of $\overline{h}$ in D_{Uni} is uniform and this is also the distribution of $\overline{h}$ that A encounters when interacting with R. Proving that, however, does not suffice to deduce that the success probability of Inverter over D_{Sim} is also high. The reason is that potentially the success probability of Inverter over D_{Uni} could stem from a relatively few elements that have significantly smaller probability mass with respect to D_{Sim} than with respect to D_{Uni}. To overcome this problem, we prove (Lemma 4.4.13) that the probability that Inverter satisfies W over D_{Uni} is well spread - even if we ignore the contribution to the success probability of some sufficiently small number of values in the support of D_{Uni}, this success probability will remain noticeable. Having that, we are guaranteed that the success probability of Inverter is high with respect to any distribution that assigns about the same mass to *most* elements in $\text{Supp}(D_{\text{Uni}})$. We then show (Lemma 4.4.14) that D_{Sim} satisfies this property.

Let us turn to a more formal discussion. For $\overline{h} \in \mathcal{H}^{m-\text{ofs}}$ we let $\varepsilon_{\overline{h}} = \Pr[A \text{ breaks the binding of } \mathsf{IH} \mid (h_1, \ldots, h_{m-\text{ofs}}) = \overline{h}]$, where $h_1, \ldots, h_{m-\text{ofs}}$ are the hash functions chosen by R in the execution of IH. We define the ***weight*** of y with respect to $\overline{h}$, by $w(y \mid \overline{h}) = \frac{1}{2}\Pr[A \text{ breaks the binding of } \mathsf{IH} \text{ outputting } ((x_0, y_0), (x_1, y_1)) \wedge y \in \{y_0, y_1\} \mid (h_1, \ldots, h_{m-\text{ofs}}) = \overline{h}]$. Note that $w(y \mid \overline{h})$ is a lower bound on the probability that Inverter satisfies W on y conditioned on $(h_1, \ldots, h_{m-\text{ofs}}) = \overline{h}$. Finally, for a given set $V \subseteq \mathcal{H}^{m-\text{ofs}} \times \{0,1\}^n$, we let $w_{\overline{V}}(y \mid \overline{h})$ be zero if $(\overline{h}, y) \in V$ and $w(y \mid \overline{h})$ otherwise.

In the following we prove two lemmata with respect to the above measures. The first lemma states that the success probability of A over D_{Uni} does not come from small sets, where the second lemma complete the picture by stating that D_{Sim} approximates D_{Uni} well over all elements in $\text{Supp}(D_{\text{Uni}})$, save but, maybe, a small set.

LEMMA 4.4.13
Let $V \subseteq \text{Supp}(D_{\text{Uni}})$ such that $\Pr\left[\left|\mathsf{Consist}(H^{m-\text{ofs}}) \cap V\right| > \sqrt{2^{\text{ofs}-1}\varepsilon_{H^{m-\text{ofs}}}}\right] < \varepsilon_A/2$, then $\mathsf{Ex}_{(\overline{h},y)\leftarrow D_{\text{Uni}}}\left[w_{\overline{V}}(y \mid \overline{h})\right] \in \Omega(\varepsilon_A 2^{m-\text{ofs}}/|L|)$.

LEMMA 4.4.14
There exists a set $V \subseteq \text{Supp}(D_{\text{Uni}})$ such that the following hold:

1. for every $(\overline{h}, y) \in \text{Supp}(D_{\text{Uni}}) \backslash V$ it holds that $\frac{1}{81} \leq \frac{D_{\text{Sim}}(\overline{h},y)}{D_{\text{Uni}}(\overline{h},y)} \leq 81$,
2. $\Pr\left[\left|\mathsf{Consist}(H^{m-\text{ofs}}) \cap V\right| > 54n^4\right] \in O(n^3 2^{m-\text{ofs}}/|L|)$.

Before proving the above lemmata, let us first use the them for proving Lemma 4.4.4. Let V be the set whose existence is guaranteed by Lemma 4.4.14. The definition of $w(\cdot)$ implies that $\Pr[\mathsf{Inverter}(\overline{h}) \in W_y] \geq w(y \mid \overline{h})$. Thus, $\Pr_{y\leftarrow L}[M^A(y) \in W_y] = \Pr_{(\overline{h},y)\leftarrow D_{\text{Sim}}}[\mathsf{Inverter}(\overline{h}) \in W_y] \geq \mathsf{Ex}_{(\overline{h},y)\leftarrow D_{\text{Sim}}}[w_{\overline{V}}(y \mid \overline{h})]$. Since D_{Uni} approximates D_{Sim} well on any element outside V, it follows

that

$$\Pr_{y\leftarrow L}[M^A(y)\in W_y]\geq \frac{1}{81}\operatorname*{Ex}_{(\overline{h},y)\leftarrow D_{\mathrm{Uni}}}[w_{\overline{V}}(y\mid \overline{h})] \tag{4.1}$$

A Markov argument yields that the success probability of A is at least $\frac{\varepsilon_A}{2}$, even if it is "forced" to fail on every $\overline{h}\in\mathcal{H}^{m-\mathrm{ofs}}$ such that $\varepsilon_{\overline{h}}<\frac{\varepsilon_A}{2}$. Thus, we can assume without loss of generality that $\sqrt{2^{\mathrm{ofs}-1}\varepsilon_{\overline{h}}}>\sqrt{2^{\mathrm{ofs}-2}\varepsilon_A}>54n^4$ for every $\varepsilon_{\overline{h}}>0$. We conclude that

$$\begin{aligned}
&\Pr\left[\left|\mathsf{Consist}(H^{m-\mathrm{ofs}})\cap V\right|>\sqrt{2^{\mathrm{ofs}-1}\varepsilon_{H^{m-\mathrm{ofs}}}}\right]\\
\leq\ &\Pr\left[\left|\mathsf{Consist}(H^{m-\mathrm{ofs}})\cap V\right|>54n^4\right]\\
\in\ &O(n^3 2^{m-\mathrm{ofs}}/\left|L\right|)<\varepsilon_A/2
\end{aligned}$$

for sufficiently large n, and the proof of Lemma 4.4.4 follows by Lemma 4.4.13 and Eq(4.1). □

Proving Lemma 4.4.13

We start by noticing that in each step of the protocol, the number of elements inside L that are consistent with the transcript so far is w.h.p. (regardless of A's answers) not faraway from the expected value.

DEFINITION 4.4.15
We call $\overline{h}\in\mathcal{H}^k$ balanced, for $k\in[m]$, if for every $j\in[k]$ it holds that $\frac{|L|}{3\cdot 2^j}\leq\left|\mathsf{Consist}(\overline{h}_{1,\ldots,j})\right|\leq\frac{3\cdot|L|}{2^j}$.

CLAIM 4.4.16
For every $k\in[m-\mathrm{ofs}]$ it holds that $\Pr[H^k \text{ is balanced}]\geq 1-6n^2 2^k/\left|L\right|$.

Proof. We say that $h\in\mathcal{H}$ ***well partitions*** the set $\mathsf{Consist}(\overline{h})$, if $\Pr_{y\leftarrow const(\overline{h})}[y\in\mathsf{Consist}(\overline{h},h)]\in[\frac{1}{2}-\frac{1}{2n},\frac{1}{2}+\frac{1}{2n}]$. Let $\overline{h}\in\mathcal{H}^k$, it is easy to verify that if $\overline{h}_j$ well partitions $\mathsf{Consist}(\overline{h}_{1,\ldots,j-1})$ for every $j\in[k]$, then $\overline{h}$ is balanced. By Lemma 2.2.3, for every $j\in[k]$ it holds that $\mathrm{P}[H \text{ does not well partition } \overline{h}_{1,\ldots,j-1}]<2\left|\mathsf{Consist}(\overline{h}_{1,\ldots,j-1})\right|/n^2$. Therefore, we can lower bound the probability that H^k is balanced as follows.

$$\begin{aligned}
&\Pr[H^k \text{ is not balanced}]\\
\leq\ &\sum_{j=1}^{k}\Pr[H^k_j \text{ does not well partition } \mathsf{Consist}(H^k_{1,\ldots,j-1})\mid H^k_{1,\ldots,j-1} \text{ is balanced}]\\
\leq\ &\sum_{j=0}^{k-1}3n^2 2^j/\left|L\right|\leq 6n^2 2^k/\left|L\right| \quad .
\end{aligned}$$

□

Having the above we are ready to prove Lemma 4.4.13.

Proof. (of Lemma 4.4.13) Let's fix for a moment $\overline{h} \in \mathcal{H}^{m-\text{ofs}}$. We assume for simplicity a non-increasing order on the elements of $\mathsf{Consist}(\overline{h})$ according to their weights (i.e., by $w(\cdot \mid \overline{h})$), and denote by $\mathsf{Consist}(\overline{h})_i$ the i^{th} element of $\mathsf{Consist}(\overline{h})$ by this order. The following claim states that the weight is not concentrated only on the first $\ell_{\overline{h}} \stackrel{\text{def}}{=} \left\lfloor \sqrt{2^{\text{ofs}-1}\varepsilon_{\overline{h}}} \right\rfloor$ heaviest elements of $\mathsf{Consist}(\overline{h})$.

CLAIM 4.4.17
It holds that $\sum_{i=\ell_{\overline{h}}+1}^{\left|\mathsf{Consist}(\overline{h})\right|} w(\mathsf{Consist}(\overline{h})_i \mid \overline{h}) \geq \varepsilon_{\overline{h}}/4$.

Proof. Let $Z = \left\{\mathsf{Consist}(\overline{h})_1, \ldots, \mathsf{Consist}(\overline{h})_{\ell_{\overline{h}}}\right\}$, by the pairwise independence of $\mathcal{H}$ it follows that,

$$\Pr[\exists y_0 \neq y_1 \in Z \ : \ H^{\text{ofs}}(y_0) = H^{\text{ofs}}(y_1)] \leq \frac{|Z|^2}{2^{\text{ofs}}} \leq \frac{2^{\text{ofs}}\varepsilon_{\overline{h}}}{2 \cdot 2^{\text{ofs}}} = \varepsilon_{\overline{h}}/2 \qquad (4.2)$$

Let y_0 and y_1 be the pair of elements returned by A on a successful cheat. Eq(4.2) yields that the probability that both y_0 and y_1 are inside Z is at most $\varepsilon_{\overline{h}}/2$. It follows that the probability that A cheats successfully while at least one of y_0 and y_1 is outside Z is at least $\varepsilon_{\overline{h}}/2$. Note that each event where A cheats successfully and outputs an element $y_i = y$, contributes half its probability to the total weight of y. Thus, the sum of weights of the elements in $\mathsf{Consist}(\overline{h}) \backslash Z$ is at least $\varepsilon_{\overline{h}}/4$. □

Assuming that $\left|\mathsf{Consist}(\overline{h}) \cap V\right| \leq \sqrt{2^{\text{ofs}-1}\varepsilon_{\overline{h}}}$, Claim 4.4.17 yields that $\sum_{y \in \mathsf{Consist}(\overline{h})} w_{\overline{V}}(y \mid \overline{h}) \geq \frac{\varepsilon_{\overline{h}}}{4}$. In the following we concentrate on the set $\text{Typical} = \left\{\overline{h} \in \mathcal{H}^{m-\text{ofs}} \ : \ \left|\mathsf{Consist}(\overline{h}) \cap V\right| \leq \sqrt{2^{\text{ofs}-1}\varepsilon_{\overline{h}}} \bigwedge \left|\mathsf{Consist}(\overline{h})\right| \leq 3\left|L\right|/2^{m-\text{ofs}}\right\}$. By Claim 4.4.16 and the assumption about V, we have that

$$\Pr[H^{m-\text{ofs}} \notin \text{Typical})] \leq \varepsilon_A/2 + O(n^2 2^{m-\text{ofs}}/\left|L\right|) \leq \frac{3 \cdot \varepsilon_A}{4} \qquad (4.3)$$

for large enough n. It follows that

$$\begin{aligned} \mathop{\mathsf{Ex}}_{(\overline{h},y) \leftarrow D_{\text{Uni}}} \left[w_{\overline{V}}(y \mid \overline{h})\right] &= \frac{1}{|\mathcal{H}^{\text{ofs}}|} \sum_{\overline{h} \in \mathcal{H}^{m-\text{ofs}}} \frac{1}{\left|\mathsf{Consist}(\overline{h})\right|} \sum_{y \in \mathsf{Consist}(\overline{h})} w_{\overline{V}}(y \mid \overline{h}) \\ \geq \ \frac{1}{|\mathcal{H}^{\text{ofs}}|} \cdot \frac{2^{m-\text{ofs}}}{3\left|L\right|} &\sum_{\overline{h} \in \text{Typical}} \sum_{y \in \mathsf{Consist}(\overline{h})} w_{\overline{V}}(y \mid \overline{h}) \ , \end{aligned}$$

and thus Claim 4.4.17 yields that

$$\mathop{\mathsf{Ex}}_{(\overline{h},y) \leftarrow D_{\text{Uni}}} \left[w_{\overline{V}}(y \mid \overline{h})\right] \geq \frac{2^{m-\text{ofs}}}{12\left|L\right|} \cdot \frac{1}{|\mathcal{H}^{\text{ofs}}|} \sum_{\overline{h} \in \text{Typical}} \varepsilon_{\overline{h}} \ .$$

Finally, using Eq(4.3) we conclude that

$$\mathop{\mathsf{Ex}}_{(\overline{h},y) \leftarrow D_{\text{Uni}}} \left[w_{\overline{V}}(y \mid \overline{h})\right] \geq \frac{2^{m-\text{ofs}}}{12\left|L\right|} \cdot \frac{\varepsilon_A}{4} \in \Omega(\varepsilon_A 2^{m-\text{ofs}}/\left|L\right|) \ .$$

□

Proving Lemma 4.4.14

We bridge between D_{Uni} and D_{Sim} using the following hybrid distributions. Let $k \in \{0,\ldots,m-1\}$ and let $\overline{h} \in \mathcal{H}^k$. We define the hybrid algorithm $\mathsf{Searcher}^{\overline{h}}(y)$ that sets its first k hash functions to $\overline{h}$ and then continues as the original $\mathsf{Searcher}$ algorithm does, and use it to define the following distributions.

- $D^{\overline{h}}_{\text{Sim}} \stackrel{\text{def}}{=} (y,h)_{y \leftarrow \mathsf{Consist}(\overline{h}), h \leftarrow \mathsf{Searcher}^{\overline{h}}(y)_{k+1}}$
- $D^{\overline{h}}_{\text{Uni}} \stackrel{\text{def}}{=} (y,h)_{h \leftarrow \mathcal{H}, y \leftarrow \mathsf{Consist}(\overline{h},h)}$

The proof of Lemma 4.4.14 easily follows by the next lemma that relates $D^{\overline{h}}_{\text{Sim}}$ to $D^{\overline{h}}_{\text{Uni}}$.

LEMMA 4.4.18

Let $k \in \{0,\ldots,m-\text{ofs}-1\}$ and let $\overline{h} \in \mathcal{H}^k$ be balanced. Then there exists a set $\text{Bad}(\overline{h}) \subseteq \mathsf{Consist}(\overline{h})$ of size at most $54n^3$ such that the following holds:

$$\Pr\left[\exists y \in \mathsf{Consist}(\overline{h},H) \setminus \text{Bad}(\overline{h}) \;:\; D^{\overline{h}}_{\text{Sim}}(y,H)/D^{\overline{h}}_{\text{Uni}}(y,H) \notin [1-\frac{4}{n},1+\frac{4}{n}]\right] \in O(n^2 2^k/|L|) \;.$$

Before proving Lemma 4.4.18, let us first use it for proving Lemma 4.4.14.

Proof. (of Lemma 4.4.14) For $\overline{h} \in \mathcal{H}^k$ let $\text{Diff}(\overline{h}) = \{y \in \mathsf{Consist}(\overline{h}) : \exists i \in [m-\text{ofs}] : D^{\overline{h}_{1,\ldots,i-1}}_{\text{Sim}}(y,\overline{h}_i)/D^{\overline{h}_{1,\ldots,i-1}}_{\text{Uni}}(y,\overline{h}_i) \notin [1-\frac{4}{n},1+\frac{4}{n}]\}$. By induction, for every $y \in \mathsf{Consist}(\overline{h}) \setminus \text{Diff}(\overline{h})$ it holds that $\frac{1}{81} \le D_{\text{Sim}}(\overline{h},y)/D_{\text{Uni}}(\overline{h},y) \le 81$. In the following we prove that w.h.p. $\text{Diff}(\overline{h})$ is small. Thus, Lemma 4.4.14 follows by letting $V = \{(y,\overline{h}) \in \text{Supp}(D_{\text{Uni}}) : y \in \text{Diff}(\overline{h})\}$.

Lemma 4.4.18 yields that $\Pr[|\text{Diff}(H^{m-\text{ofs}})| > 54n^4 \mid H^{m-\text{ofs}} \text{ is balanced}] \le n \cdot \Omega(n^2 2^k/|L|)$. Where by Claim 4.4.16 we have that $\Pr[H^{m-\text{ofs}} \text{ is balanced}] \ge 1 - \frac{6n^2 2^{m-\text{ofs}}}{|L|}$. It follows that $\Pr[|\text{Diff}(H^{m-\text{ofs}})| > 54n^4] \le n \cdot \Omega(n^2 2^k/|L|) + \frac{6n^2 2^{m-\text{ofs}}}{|L|} \in \Omega(n^3 2^{m-\text{ofs}}/|L|)$. □

Proof. (of Lemma 4.4.18) Consider the Boolean matrix $T^{|\mathsf{Consist}(\overline{h})| \times |\mathcal{H}|}$, where $T(y,h) = 1$ if $A^{Com}(\overline{h},h)_{k+1} = h(y)$ and zero otherwise. We identify the indices into T with the set $\mathsf{Consist}(\overline{h}) \times \mathcal{H}$. The distribution $D^{\overline{h}}_{\text{Uni}}$ can be described in relation to T as follows. Choose a random column of T and draw the index of a random one entry from this column (where a "one entry" is simply an entry of the matrix that is assigned the value one). The distribution $D^{\overline{h}}_{\text{Sim}}$ can also be described in relation to T as follows. Choose a random row of T and for $2\log(n)$ times draw a random entry from this raw. If a one entry is drawn, then choose its index and stop drawing, otherwise select the index of the last drawn entry.

Let us start with an informal discussion. Compare the matrix T with the matrix $\widehat{T}^{|\mathsf{Consist}(\overline{h})| \times |\mathcal{H}|}$, where $\widehat{T}(y,h) = h(y)$. Note that T can be derived from $\widehat{T}$ by flipping all values in some of its columns (where the column which corresponds to h is flipped whenever $A^{Com}(\overline{h},h)_{k+1} = 0$). By the pairwise independence of $\mathcal{H}$, it follows that most *columns* of $\widehat{T}$ are balanced (have about the same number of zeros and ones) and thus the same holds for T. Hence, the mass that $D^{\overline{h}}_{\text{Uni}}$ assigns to most of the one entries of T is close to $\frac{1}{|\mathcal{H}|} \cdot \frac{2}{|\mathsf{Consist}(\overline{h})|}$. Using again the pairwise independent of

$\mathcal{H}$, we can prove that most *rows* of T are balanced. Hence, the mass that $D^{\overline{h}}_{\text{Sim}}$ assigns to most one entries in T is also close to $\frac{1}{|\mathcal{H}|} \cdot \frac{2}{|\mathsf{Consist}(\overline{h})|}$. Since the support of $D^{\overline{h}}_{\text{Uni}}$ and the indices set of one entries in T are the same, we conclude that the one indices in a random raw of T (a random $h \in \mathcal{H}$) get about the same mass in $D^{\overline{h}}_{\text{Sim}}$ and in $D^{\overline{h}}_{\text{Uni}}$, and the proof of the Lemma 4.4.18 follows.

Let us turn to the formal proof. We define the set $\text{Bad}(\overline{h})$ as $\{y \in \mathsf{Consist}(\overline{h}) : \Pr[T(H,y)=1] \notin [\frac{1}{2}-\frac{1}{2n}, \frac{1}{2}+\frac{1}{2n}]\}$ and start by showing that $\text{Bad}(\overline{h})$ is indeed small.

CLAIM 4.4.19
Assuming that $\overline{h}$ is balanced, then $\left|\text{Bad}(\overline{h})\right| < 54n^3$.

Proof. Let $\text{Bad}_{Law}(\overline{h}) = \{y \in \mathsf{Consist}(\overline{h}) : \Pr[T(H,y)=1] < \frac{1}{2}-\frac{1}{2n}\}$. We assume that $\left|\text{Bad}_{Law}(\overline{h})\right| > 27n^3$ and derive a contradiction (the proof that $\left|\text{Bad}(\overline{h}) \setminus \text{Bad}_{Law}(\overline{h})\right| < 27n^3$ is analogous). Consider the matrix $T|_{\text{Bad}_{Law}}$ - the restriction of T to the rows $\text{Bad}_{Law}(\overline{h})$. By definition, the rows of $T|_{\text{Bad}_{Law}}$ have more zeros than ones. Hence, the matrix $T|_{\text{Bad}_{Law}}$ itself has more zeros than ones. On the other hand, by the pairwise independence of $\mathcal{H}$ it follows that most columns of $T|_{\text{Bad}_{Law}}$ are balanced (have about the same number of zeros of ones). Therefore, $T|_{\text{Bad}_{Law}}$ itself is balanced and a contradiction is derived. More formally, for $h \in \mathcal{H}$ let T_h be the number of ones in the h column, that is $T_h = \sum_{y \in \text{Bad}_{Law}(\overline{h})} T(y,h)$. We upper bound the expectation of T_h as follows,

$$\mathsf{Ex}[T_H] = \mathsf{Ex}\Big[\sum_{y \in \text{Bad}_{Law}(\overline{h})} T(y,H)\Big] = \sum_{y \in \text{Bad}_{Law}(\overline{h})} \mathsf{Ex}[T(y,H)] < (\frac{1}{2}-\frac{1}{2n})\left|\text{Bad}_{Law}(\overline{h})\right| \ .$$

Recall that $T(h,y)=1$ if $A^{Com}(\overline{h},h)_{k+1} = h(y)$ and zero otherwise. Since the set $\text{Bad}_{Law}(\overline{h})$ is large, Lemma 2.2.3 yields that a random h splits w.h.p. the elements of $\text{Bad}_{Law}(\overline{h})$ into two almost equals size according to their consistency with A's answer on h. That it, $\Pr\left[T_H < \left|\text{Bad}_{Law}(\overline{h})\right| \cdot (\frac{1}{2}-\frac{1}{3n})\right] < \frac{9n^2}{\left|\text{Bad}_{Law}(\overline{h})\right|} \leq \frac{1}{3n}$. Thus,

$$\begin{aligned}\mathsf{Ex}[T_H] &\geq \frac{1}{|H|} \cdot \sum_{h \in \mathcal{H} \ : \ T_h \geq \left|\text{Bad}_{Law}(\overline{h})\right| \cdot (\frac{1}{2}-\frac{1}{3n})} \left|\text{Bad}_{Law}(\overline{h})\right| \cdot (\frac{1}{2}-\frac{1}{3n}) \\ &> (1-\frac{1}{3n}) \cdot \left|\text{Bad}_{Law}(\overline{h})\right| \cdot (\frac{1}{2}-\frac{1}{3n}) > \left|\text{Bad}_{Law}(\overline{h})\right| \cdot (\frac{1}{2}-\frac{1}{2n}) \ ,\end{aligned}$$

and a contradiction is derived. □

The definition of $\text{Bad}(\overline{h})$ yields that $\Pr_{(y,h) \leftarrow D^{\overline{h}}_{\text{Sim}}}[A^{Com}(\overline{h},h)_{k+1} = h(y) \mid y \notin \text{Bad}(\overline{h})] > 1 - O(1/n^2)$. Thus for every $h \in \mathcal{H}$ and every $y \in \mathsf{Consist}(\overline{h},h) \setminus \text{Bad}(\overline{h})$, it holds that $D^{\overline{h}}_{\text{Sim}}(y,h) \in [(1-O(1/n^2))\gamma \cdot \frac{1}{1+\frac{1}{n}}, \gamma \cdot \frac{1}{1-\frac{1}{n}}]$, for $\gamma = \frac{2}{|\mathsf{Consist}(\overline{h})| \cdot |\mathcal{H}|}$. Now let $\mathcal{H}^{\text{Bad}}(\overline{h}) = \{h \in \mathcal{H} : \Pr_{y \leftarrow \mathsf{Consist}(\overline{h})}[T(h,y)=1] \notin [\frac{1}{2}-\frac{1}{2n}, \frac{1}{2}+\frac{1}{2n}]\}$. Clearly for every $h \in \mathcal{H} \setminus \mathcal{H}^{\text{Bad}}(\overline{h})$ and every $y \in \mathsf{Consist}(\overline{h},h)$, it holds that $D^{\overline{h}}_{\text{Uni}}(y,h) \in [\gamma \cdot \frac{1}{1+\frac{1}{n}}, \gamma \cdot \frac{1}{1-\frac{1}{n}}]$. It follows that $D^{\overline{h}}_{\text{Sim}}(y,h)/D^{\overline{h}}_{\text{Uni}}(y,h) \in [1-\frac{4}{n}, 1+\frac{4}{n}]$ for every $h \in \mathcal{H} \setminus \mathcal{H}^{\text{Bad}}(\overline{h})$ and $y \in \mathsf{Consist}(\overline{h},h) \setminus \text{Bad}(\overline{h})$, and the proof of Lemma 4.4.18 follows by next claim.

CLAIM 4.4.20
Assuming that $\overline{h}$ is balanced, then $\Pr[H \in \mathcal{H}^{\mathrm{Bad}}(\overline{h})] \in \Omega(n^2 2^k / |L|)$.

Proof. Immediate by the pairwise independence of $\mathcal{H}$ (see Lemma 2.2.3) □

□

4.5 Protocols with Better Round Complexity

When Protocol 4.4.2 is invoked with Boolean hash functions (as in the proof of Theorem 4.4.1), it suffers from a linear round complexity. Unfortunately, it seems that such a round complexity is unavoidable for interactive hashing protocols that are both hiding and binding (for more details, see the discussion at the end of Section 4.1.3). Having the above, we consider more efficient protocols whose binding hold with respect to relations that are more than super-polynomial hard. In particular, we consider Protocol 4.4.2 invoked with hash functions whose output length is roughly "log the security" of a given relation, and prove the following lemma that generalizes Lemma 4.4.4.

LEMMA 4.5.1
Let W be a binary relation and let $L \subseteq \{0,1\}^n$. Let $\mathcal{H}$ be an efficient family of pairwise independent hash functions from strings of length n to strings of length $s(n)$ and let $m(n) \leq \left\lfloor \frac{\log(|L|)}{s(n)} \right\rfloor$. Finally, let IH be the instantiation of Protocol 4.4.2 with $\mathcal{H}$ and m, and let A be an algorithm that runs in time $t_A(n)$ and breaks the binding of IH with respect to W and L with probability $\varepsilon_A(n)$.

Then there exists an oracle algorithm $M^{(\cdot)}$ that given an oracle access to A, the following holds for large enough n.

$$\Pr_{y \leftarrow L}[M^A(y) \in W_y] \in \Omega\left(\frac{2^{m(n)}}{|L|} \cdot \frac{\varepsilon_A(n)^2}{2^{2s(n)} n^8}\right) .$$

Letting $t_{\mathcal{H}}(n)$ be an upper bound of the sampling and computing time of $\mathcal{H}$, the running-time of M^A is $O(\log(n) 2^{s(n)}(t_A(n) + m(n) t_{\mathcal{H}}(n)))$.

REMARK 4.5.2
As in Lemma 4.4.4, the binding for $m(n) > \left\lfloor \frac{\log(|L|)}{s(n)} \right\rfloor$ immediately follows by the binding of the first $\left\lfloor \frac{\log(|L|)}{s(n)} \right\rfloor$ rounds. In addition, M^A does not need to know L, W or ε_A.

Proof's sketch. The proof of Lemma 4.5.1 follows very closely the proof of Lemma 4.4.4 and we only point out the main differences. When using hash functions of $s(n)$ bit output rather than Boolean ones, each query made by the receiver partitions the set of consistent elements into $2^{s(n)}$ different subsets (rather than two subsets as in the Boolean case). For the second part of the proof to go through (i.e., Lemma 4.4.14), we need to make sure that the size of each of the subsets induced by a random query is not too far from the expected value. Since we want to use pairwise independence concentration laws (and in particular, Lemma 2.2.3), we need to make sure the initial set of consistent elements is large enough. For the same reason, we cannot guarantee that the set of "bad" elements for which D_{Uni} and D_{Sim} give different weight (i.e., the set V in Lemma 4.4.14) is very small, but rather have to compensate a much larger set of about $O(n^4 2^{s(n)})$

elements. It turns out that we can adopt Lemma 4.4.14 for hash functions that output $s(n)$ bits and V of size $O(n^4 2^{s(n)})$, by taking ofs $\in \Omega\left(\log(n) + \log(1/\varepsilon_A) + 2s(n)\right)$ and change the main loop of the Searcher algorithm (where Searcher tries to find a hash function h such that A's answer on h is equal to $h(y)$) to repeat $2\log(n) \cdot 2^{s(n)}$ times (rather than $2\log(n)$ times). That is, we get the following lemma.

LEMMA 4.5.3

There exists a set $V \subseteq \mathrm{Supp}(D_{\mathrm{Uni}})$ such that the following hold:

1. for every $(\overline{h}, y) \in \mathrm{Supp}(D_{\mathrm{Uni}}) \backslash V$ it holds that $\frac{1}{81} \leq \frac{D_{\mathrm{Sim}}(\overline{h},y)}{D_{\mathrm{Uni}}(\overline{h},y)} \leq 81$,
2. $\Pr\left[\left|\mathsf{Consist}(H^{m-\mathrm{ofs}}) \cap V\right| > 54n^4 2^{s(n)}\right] \in O(n^3 2^{m-\mathrm{ofs}}/\left|L\right|)$,

where D_{Uni} and D_{Sim} are as in Definition 4.4.12 but with respect to ofs $= \max\left\{m \cdot s(n), \lceil 8\log(n) + \log(1/\varepsilon_A)\rceil + 13 + 2s(n)\right\}$.

In order to complete the proof of Lemma 4.5.1, it is left to show that we can apply Lemma 4.4.13 with the bound for $|V|$ promised by Lemma 4.5.3 and the new value of ofs. While Lemma 4.4.13 is stated for Boolean hash functions, it is easy to verify that its proof does not use the fact that the (pairwise independent) hash functions are Boolean rather than an arbitrary sequence of hash functions with overall output length ofs. Thus, we can use Lemma 4.4.13 for non Boolean hash functions, and the proof of Lemma 4.5.1 goes through.

4.6 Applying Our New Proof to NOVY

In this section we show the proof of Lemma 4.4.4 can be applied to the following protocol, known as the NOVY protocol, considered by Naor et al. [NOVY98] and Nguyen et al [NOV06].

4.6.1 The NOVY protocol

For $m(n) \in \mathbb{N}$, we define the following protocol.

PROTOCOL 4.6.1

The NOVY protocol $\mathsf{IH} = (\mathsf{S}, \mathsf{R})$.

Common input: 1^n.

S's inputs: $y \in \{0,1\}^n$.

1. For $i = 1$ to $m(n)$:
   (a) R chooses uniformly at random $r_i \in \{0,1\}^{n-i}$ and sends $h_i = 0^{i-1} \circ 1 \circ r_i$ over to S.
   (b) S sends $z_i = \langle h_i, y\rangle_2 \mod 2$ back to R.
2. S locally outputs y.
3. R outputs $(\overline{h}, \overline{z}) = ((h_1, \ldots, h_{m(n)}), (z_1, \ldots, z_{m(n)}))$.

That is, the above protocol is the same as Protocol 4.4.2, but it uses a special type of Boolean function. For $i \in \{1,\ldots,m(n)\}$ let $\mathcal{H}_i$ be the family of functions induced by the selection of h_i described above.

4.6.2 The New Lemma

Our goal is to prove the following version of Lemma 4.4.4.

LEMMA 4.6.2
Let W be a binary relation, let $m(n) \leq n$ and let (S,R) and $\{\mathcal{H}_i\}_{i=1}^{m(n)}$ be as in Protocol 4.6.1. Finally, let A be an algorithm that runs in time $t_A(n)$ and breaks the binding of (S,R) with respect to W with probability $\varepsilon_A(n)$. Then there exists an oracle algorithm $M^{(\cdot)}$ that given an oracle access to A, the following holds for large enough n.

$$\Pr[M^A(U_n) \in W_{U_n}] \in \Omega(\tfrac{1}{2^{n-m(n)}} \cdot \tfrac{\varepsilon_A(n)^2}{n^8}) \ .$$

The running-time of M^A is $O(n\log(n) + mn\log(n))$.

Proof. It is easy to verify that if the families of functions $\{\mathcal{H}_i\}_{i=1}^{n-1}$ would have been pairwise independent, then our proof of Lemma 4.4.4 would hold in this case as well.[14] The latter, however, does not hold and therefore we have to refine our approach. Fortunately, the proof of the theorem does not require that the families of Boolean hash function to be pairwise independent with respect to the initial set of inputs L, but rather to be pairwise independent with respect to the elements of the initial set that are consistent with the protocol so far. It turns out that given that the initial set is $\{0,1\}^n$, the families of Boolean hash functions used by NOVY are "pairwise independent enough" on the relevant set and thus essentially the same proof as the one we gave for Lemma 4.4.4 goes through.

Let's us turn to a more formal discussion. For $k \in [m(n)]$, $\overline{h} \in \mathcal{H}_1 \times \cdots \times \mathcal{H}_k$ and $\overline{z} \in \{0,1\}^k$, let $\mathsf{Consist}(\overline{h},\overline{z})$ be the set of elements inside $\{0,1\}^n$ that are consistent with $\overline{h}$ and $\overline{z}$ (i.e., $\left\{y \in \{0,1\}^n \ : \ \overline{h}(y) = \overline{z}\right\}$). By induction, it follows that for every possible pair $(\overline{h},\overline{z})$ and element $y_2 \in \{0,1\}^{n-k}$, there exists a *single* element $y_1 \in \{0,1\}^k$ (which depends on y_2 and on $(\overline{h},\overline{z})$) such that $y_1 \circ y_2 \in \mathsf{Consist}(\overline{h},\overline{z})$. Hence, for any $y \in \mathsf{Consist}(\overline{h},\overline{z})$ there exists exactly one other element $y' \in \mathsf{Consist}(\overline{h},\overline{z})$ for which $y_{k+2\ldots,n} = y'_{k+2\ldots,n}$. Thus, for any other element $y'' \in \mathsf{Consist}(\overline{h},\overline{z})$, which is different than y and y', it holds that the random variables $h(y)$ and $h(y'')$ (and also $h(y')$ and $h(y'')$), where h is a random function from $\mathcal{H}_{k+1}$, are independent. Therefore, every subset $Z \subseteq \mathsf{Consist}(\overline{h},\overline{z})$ can be partitioned into two almost equal size subsets (i.e., of difference in size at most one) such that $\mathcal{H}_{k+1}$ is pairwise independent with respect to both subsets. Through the proof of Lemma 4.4.4, we use the pairwise independence property of the hash functions to prove that the following holds. Every fixed large enough subset $Z \subseteq \mathsf{Consist}(\overline{h},\overline{z})$ is partitioned w.h.p. by a random Boolean hash function into two parts of almost the same size.[15] By our previous

[14] Note that the proof of Lemma 4.4.4 does not require that the same family is used in each round.

[15] Actually, save but the proof of Claim 4.4.19, we only need this property with respect to $Z = \mathsf{Consist}(\overline{h},\overline{z})$. Note that by the above observation about the structure of $\mathsf{Consist}(\overline{h},\overline{z})$, every $h \in \mathcal{H}_{k+1}$ *always* partitions $\mathsf{Consist}(\overline{h},\overline{z})$ into two equal parts.

observation such a partition also happens, with high enough probability, with respect to the family $\mathcal{H}_{k+1}$. Thus, the proof of Lemma 4.4.4 applied also for the NOVY protocol. □

4.7 Conclusions

One interesting question is to come with a reduction from breaking the binding of an interactive hashing to violating the hardness of the underlying relation that is even more security preserving that the one given in Lemma 4.4.4. Particularly, is there such a reduction that is linearly-preserving [HL92] (i.e., where the time-success ratio of an adversary violating the hardness of the relation is only larger by a multiplicative polynomial factor than the time-success ratio of an adversary breaking the binding of the interactive hashing protocol). There are three possible directions for an improvement: (1) Presenting a more secure protocol than Protocol 4.4.2, (2) Giving a better reduction from an adversary that breaks the interactive hashing to one that violates the hardness of the relation, or (3) Improving the analysis of the reduction mentioned in (2).

We mention that our improvement in parameters over the NOVY proof is mainly in the third item (i.e., the analysis of the reduction). In the following we show that our analysis cannot be pushed much further. Namely, we present a (non-efficient) adversary A that breaks the binding of Protocol 4.4.2 (invoked with Boolean pairwise independence hash functions) with probability ε, but M^A breaks the underlying relation (in this case the relation imposed by a one-way permutation) with probability at most $2 \cdot \varepsilon^{1.4}$.

Consider an algorithm M for inverting a one-way permutation that uses an adversary A of Protocol 4.4.2 (or of the NOVY protocol) in the following black-box manner: on $y \in \{0,1\}^n$, it keeps sampling random hash functions and rewinding A, until it finds a series of $n-1$ hash functions on which A's answers is consistent with y. Then, it returns one of A's outputs as the candidate preimage of y (note that both the NOVY and ours inverting algorithms follow this strategy). Assume that A operates as follows: for $\varepsilon > 0$, it replies with random answers on the first $n - \log(\frac{1}{\varepsilon})$ questions (hash functions) and then randomly selects two distinct elements, $y_1, y_2 \in \{0,1\}^n$, that are consistent with the protocol so far. For the remaining hash functions A does the following: if both y_1 and y_2 yield the same answer then it answers with this value, otherwise, it selects randomly one of the elements and from now on answers according to this element. At the end of the protocol A checks whether both y_1 and y_2 are consistent with the protocol. If the answer is positive, it inverts f on both y_1 and y_2 and outputs the result (recall that the reduction does not assume that A is efficient and therefore it is allowed for example to invert f using exhaustive search), otherwise it outputs $\perp$. Since $\mathcal{H}$ is a family of pairwise independent hash functions, the random variables $h(y_1)$ and $h(y_2)$, for a randomly chosen hash function h, are independent.[16] Thus, the probability that A breaks the binding of Protocol 4.4.2 is exactly ε. On the other hand, in order for M to succeed, y has to be selected by A as one of the elements in $\{y_1, y_2\}$. Since the number of elements the are consistent with the protocol after $n - \log(\frac{1}{\varepsilon})$ steps is $1/\varepsilon$, it follows that this happens with probability 2ε. Given that $y \in \{y_1, y_2\}$, say that $y = y_1$, M has to choose in each step an hash function h for which $A(h) = h(y) = h(y_2)$. By the independence of $h(y)$ and $h(y_2)$, it follows that the probability that $A(h) = h(y) \neq h(y_2)$ is exactly $\frac{1}{4}$. Therefore, the probability that in all the last $\log(\frac{1}{\varepsilon})$ steps it holds that $A(h) = h(y) = h(y_2)$, is at most $(\frac{3}{4})^{\log(\frac{1}{\varepsilon})} < \varepsilon^{0.4}$. We conclude that the overall success probability of M^A is at most $2 \cdot \varepsilon^{1.4}$.

[16] As mentioned in Section 4.6, the hash functions used by the NOVY protocol are not exactly pairwise independent. However, almost the same argument holds for the NOVY hash functions.

In the above case, it is easy to present an algorithm that inverts the one-way permutation, using black-box access to A, with probability that is very close to ε. Nevertheless, it is possible that one can generalize and strengthen the above argument to preclude any linearly-preserving black-box reduction from interactive hashing to violating the underlying relation. Such a separation would be quite informative (an easier task would be to rule out any black-box proof that the binding of Protocol 4.4.2 is linearly preserving).

4.8 A Simpler Construction of Statistically Hiding Commitment from Known Regular One-way Functions

In this section we use our new interactive hashing theorem (Theorem 4.4.1) to give a direct construction of statistically hiding commitment from known regular one-way functions.

A commitment scheme is a two-stage protocol between a sender and a receiver. In the first stage, called the ***commit stage***, the sender commits to a private string σ. In the second stage, called the ***reveal stage***, the sender reveals σ and *proves* that it was the value to which she committed in the first stage. We require two properties of commitment schemes. The hiding property says that the receiver learns nothing about σ in the commit stage. The binding property says that after the commit stage, the sender is bound to a particular value of σ; that is, she cannot successfully open the commitment to two different values in the reveal stage. In a statistically hiding and computationally binding commitment scheme, the hiding holds information theoretically (i.e., even an all powerful learns nothing about σ), where the binding property only guaranteed to hold against polynomial-time senders. A known regular one-way function is an efficiently computable function that is hard to invert, and all its images have the same (efficiently computable) number of preimages. For the formal definitions of these primitives, see for example [HHK+05].

Our construction is achieved by applying an interactive hashing protocol to the output of the one-way function. The new construction somewhat simplifies the previous construction, and proof, of Haitner et al. [HHK+05, Theorem 4.4], which uses an additional hashing step before applying the NOVY protocol.

THEOREM 4.8.1
If there exists known regular one-way functions, then there exists statistically hiding and computationally binding commitment schemes.

REMARK 4.8.2
[HHK+05, Theorem 4.4] also holds with respect to somewhat more general choice of one-way functions.[17] It is easy to verify, however, that the proofs given here can be extended also to these settings.

Proof. (of Theorem 4.8.1) We use our new interactive hashing theorem to get a bit commitment scheme (i.e., the committed string is a single bit) that is "somewhat hiding" in the meaning define below, and the existence of a full-fledge commitment scheme follows via standard amplification techniques.

[17] Informally, [HHK+05] consider the case where the number of preimages is not fixed for all outputs, but rather can be efficiently approximated.

DEFINITION 4.8.3 (weakly hiding commitment scheme)
Let $\delta : \mathbb{N} \mapsto \mathbb{R}^+$. A commitment scheme $\mathsf{Com} = (\mathsf{S} = (\mathsf{S}_c, \mathsf{S}_r), \mathsf{R} = (\mathsf{R}_c, \mathsf{R}_r))$ is statistically $\delta(n)$-hiding, if for every algorithm R^* the ensembles $\{\text{view}_{\mathsf{R}^*}(\mathsf{S}_c(0), \mathsf{R}^*)(1^n)\}_{n\in\mathbb{N}}$ and $\{\text{view}_{\mathsf{R}^*}(\mathsf{S}_c(1), \mathsf{R}^*)(1^n)\}_{n\in\mathbb{N}}$ are of statistical distance at most $\delta(n)$, where $\text{view}_{\mathsf{R}^*}(\mathsf{S}_c(b), \mathsf{R}^*)$ denotes the view of R^* in the commit stage interacting with $\mathsf{S}_c(b)$.

Let $f\colon \{0,1\}^n \mapsto \{0,1\}^n$ be a regular one-way function[18] and let $W = \{(x, f(x))\colon x \in \{0,1\}^n\}$. We first note that the regularity of f implies that the distributions $f(U_n)$ and the uniform distribution over $Im(f)$ are the same. Since f is hard to invert over $f(U_n)$, it follows that W is hard to satisfy over $Im(f)$. For $m = \lfloor \log |Im(f)| \rfloor - 9$, let $\mathcal{H}$ and $\mathsf{IH} = (\mathsf{S}^I, \mathsf{R}^I)$ be the hash family and interactive hashing protocol guarantees by Theorem 4.4.1 for the above W and $L = Im(f)$. We use the following bit commitment protocol.

4.8.1 The Weakly Hiding Protocol

Let $\mathcal{G}$ be an efficient family of Boolean pairwise independent hash functions defined over strings of length n and let f, $\mathcal{H}$ and IH be as above. We define the bit commitment protocol $\mathsf{Com} = (\mathsf{S}, \mathsf{R})$ as follows:

PROTOCOL 4.8.4
The commitment scheme $\mathsf{Com} = (\mathsf{S} = (\mathsf{S}_c, \mathsf{S}_r), \mathsf{R} = (\mathsf{R}_c, \mathsf{R}_r))$.

Commit stage.

Common input: 1^n.

S_c's input: $b \in \{0,1\}$.

1. S_c chooses uniformly at random $x \in \{0,1\}^n$ and sets $y = f(x)$.
2. $(\mathsf{S}_c, \mathsf{R}_c)$ runs $(\mathsf{S}^I(y, 1^n), \mathsf{R}^I(1^n))$, with S_c and R_c acting S^I and R^I respectively. Let (h, z) be the output of R^I in this execution.
3. S_c chooses uniformly at random $g \in \mathcal{G}$ and sends g, $c = b \oplus g(y)$ to R.
4. S_c locally outputs x and R_c outputs (h, z, g, c).

Reveal stage.[19]

Common input: 1^n, $b \in \{0,1\}$ and (h, z, g, c).

S_r's input: x.

1. S_r sends x to R_r.

[18] The assumption that f is length-preserving is without lost of generality, see [Gol01a, HHR06].

[19] Alternatively, we could have used the generic reveal stage - S_r sends the random-coins of S_c and R_r checks for consistency.

2. R_r accepts if $h(f(x)) = z$ and $g(f(x)) \oplus b = c$.

In Lemma 4.8.5 we show that Protocol 4.8.4 is computationally binding, where Lemma 4.8.6 states is (statistically) C-hiding for some constant $C > 0$. Thus, the proof of Theorem 4.8.1 follows by standard amplification techniques (e.g., [HHK+05, Thm. 5.2]). □

LEMMA 4.8.5
Protocol 4.8.4 is computationally binding.

Proof. We show that any adversary that breaks the binding of the bit-commitment protocol can be trivially used to break the binding of the underlying interactive hashing protocol IH. Specifically, given an adversary A that breaks the binding of the bit-commitment with non-negligible probability, the following algorithm, M^A, uses A to break the binding of IH. Algorithm M^A acts as A in the interaction with R_I, let (h, z) be the public output of R_I that follows this interaction. Note that the only interaction of A with R is the interaction with R_I, therefore from A's point of view it has just took part in a normal execution of $(\mathsf{S}_c, \mathsf{R}_c)$. Let (g, c) be A's message that follows this interaction. By the contradiction assumption, in the reveal stage A outputs with non-negligible probability two elements x_0 and x_1 such that for both $i \in \{0, 1\}$ it holds that $h(f(x_i)) = z$ and $g(f(x_i)) \oplus i = c$. In particular, it holds that $h(f(x_0)) = h(f(x_1)) = z$ and $f(x_0) \neq f(x_1)$. Thus, by outputting $(x_0, f(x_0))$ and $(x_1, f(x_1))$, M^A breaks the binding of IH. □

LEMMA 4.8.6
Protocol 4.8.4 is statistically $\frac{7}{8}$-hiding.

Proof. Let R^* be an adversary playing the role of R_c in Com, for $b \in \{0, 1\}$ let $V_{\mathsf{R}^*}(b)$ denote R^*'s view in the interaction with $\mathsf{S}_c(b)$. We split $V_{\mathsf{R}^*}(b)$ into two parts: the interaction of R^* with S^I and the last message of S_c (i.e., (g, c), which may be empty), and denote these random variables by V_{IH} and $V_{\mathcal{G}}(b)$ respectively (note that V_{IH} is independent of b). Using the weakly-hiding property of IH, we show that with sufficient probability V_{IH} is consistent with many y's. Thus, g partitions the set of consistent y's into two subsets of similar size. Since a commitment to 0 when y is taken from one of these subsets generates the same view as a commitment to 1 when y is taken from the other subset, it follows that the views in both cases are statistically close. Namely, the protocol is weakly hiding.

Let us turn to the formal proof. The weakly-hiding property of IH yields the existence of algorithms Ext and Sim such that for both $b \in \{0, 1\}$ it holds that $(\mathsf{Sim}^{\mathsf{R}^*}(H(Y)), V_{\mathcal{G}}(b))$ and $V_{\mathsf{R}^*}(b)$ have the same distribution, where Y is the value of y $(= f(x))$ selected by S_c and $H = \mathrm{Ext}(V_{\mathsf{IH}})$. In the following we fix the random-coins of S^I, R^*, Ext and Sim (but not the values of y and g) and prove that $(\mathsf{Sim}^{\mathsf{R}^*}(H(Y)), V_{\mathcal{G}}(0))$ and $(\mathsf{Sim}^{\mathsf{R}^*}(H(Y)), V_{\mathcal{G}}(1))$ are statistically close. Thus, for every fixing $V_{\mathsf{R}^*}(0)$ and $V_{\mathsf{R}^*}(1)$ are statistically close, and the proof of the lemma follows.

Fix the random-coins of S^I, R^*, Ext and Sim, note that under this fixing $\mathsf{Sim}^{\mathsf{R}^*}(H(Y))$ is determined by Y. For $z \in \mathsf{Sim}^{\mathsf{R}^*}(H(Y))$, let $\mathsf{Consist}(z)$ be the set of $y \in Im(f)$ that are consistent with z (i.e., conditioned on $Y = y$ it holds that $\mathsf{Sim}^{\mathsf{R}^*}(H(Y)) = z$). The following holds for every

$z \in Sim^{\mathsf{R}^*}(H(Y))$:

$$\begin{aligned}
& \Delta(V_{\mathsf{R}^*}(0), V_{\mathsf{R}^*}(1) \mid \mathsf{Sim}^{\mathsf{R}^*}(H(Y)) = z) \qquad (4.4)\\
= \;& \Delta((z, V_{\mathcal{G}}(0)), (z, V_{\mathcal{G}}(1)) \mid \mathsf{Sim}^{\mathsf{R}^*}(H(Y)) = z)\\
\leq \;& \frac{1}{2} \sum_{g \in \mathcal{G}, c \in \{0,1\}} \left| \Pr[V_{\mathcal{G}}(0) = (g,c) \mid \mathsf{Sim}^{\mathsf{R}^*}(H(Y)) = z] - \Pr[V_{\mathcal{G}}(1) = (g,c) \mid \mathsf{Sim}^{\mathsf{R}^*}(H(Y)) = z] \right|\\
= \;& \frac{1}{2\,|\mathcal{G}|} \sum_{g \in \mathcal{G}, c \in \{0,1\}} \left| \Pr_{y \leftarrow \mathsf{Consist}(z)}[g(y) \oplus 0 = c] - \Pr_{y \leftarrow \mathsf{Consist}(z)}[g(y) \oplus 1 = c] \right| ,
\end{aligned}$$

where the inequality is since S might abort in z, which yields that $V_{\mathcal{G}}(0)$ and $V_{\mathcal{G}}(0)$ are the empty message. Using the pairwise independence of $\mathcal{G}$ (Lemma 2.2.3, letting $\ell = 1$, $L = \mathsf{Consist}(z)$ and $\delta = \frac{1}{2}$) we have that $\Pr_{g \leftarrow \mathcal{G}}\left[\Pr_{y \leftarrow \mathsf{Consist}(z)}[g(y) = 0] \notin [\frac{1}{4}, \frac{3}{4}]\right] < \frac{8}{|\mathsf{Consist}(z)|}$, which together with Eq(4.4) yield that

$$\Delta(V_{\mathsf{R}^*}(0), V_{\mathsf{R}^*}(1) \mid \mathsf{Sim}^{\mathsf{R}^*}(H(Y)) = z) \leq \frac{16}{|\mathsf{Consist}(z)|} + \frac{1}{2} \qquad (4.5)$$

We call $z \in \mathsf{Sim}^{\mathsf{R}^*}(H(Y))$ ***heavy*** if $|\mathsf{Consist}(z)| \geq 64$. By Eq(4.5) we have that $\Delta(V_{\mathsf{R}^*}(0), V_{\mathsf{R}^*}(1) \mid \mathsf{Sim}^{\mathsf{R}^*}(H(Y)) = z) \leq \frac{3}{4}$ for every heavy z, and we conclude the proof by proving that $\mathsf{Sim}^{\mathsf{R}^*}(H(Y))$ is heavy with high probability.

CLAIM 4.8.7
$\Pr[\mathsf{Sim}^{\mathsf{R}^*}(H(Y)) \text{ is heavy}] \geq \frac{7}{8}$.

Proof. For $y \in Im(f)$ let $t(y)$ be the value of $H(Y)$ conditioned that $Y = y$. A simple counting argument (recall that $m = \lfloor \log |Im(f)| \rfloor - 9$) yields that $\Pr_{y \leftarrow Im(f)}[t^{-1}(t(y)) \geq 64] \geq \frac{7}{8}$, and the claim follows since $\mathsf{Sim}^{\mathsf{R}^*}(H(Y))$ is a deterministic function of $t(Y)$. □

Thus,

$$\begin{aligned}
& \Delta(V_{\mathsf{R}^*}(0), V_{\mathsf{R}^*}(1))\\
= \;& \Pr[\mathsf{Sim}^{\mathsf{R}^*}(H(Y)) \text{ is heavy}] \cdot \Delta(V_{\mathsf{R}^*}(0), V_{\mathsf{R}^*}(1) \mid \mathsf{Sim}^{\mathsf{R}^*}(H(Y)) \text{ is heavy})\\
+ \;& \Pr[\mathsf{Sim}^{\mathsf{R}^*}(H(Y)) \text{ is not heavy}] \cdot \Delta(V_{\mathsf{R}^*}(0), V_{\mathsf{R}^*}(1) \mid \mathsf{Sim}^{\mathsf{R}^*}(H(Y)) \text{ is not heavy})\\
\leq \;& \frac{3}{4} + \Pr[\mathsf{Sim}^{\mathsf{R}^*}(H(Y)) \text{ is not heavy}] < \frac{7}{8}
\end{aligned}$$

□

Chapter 5

Efficient Pseudorandom Generators

5.1 Introduction

In this work we address two fundamental problems in cryptography: (1) constructing pseudorandom generators from one-way functions and (2) transforming weak one-way functions into strong one-way functions. The common thread linking the two problems in our discussion is the technique we use. This technique that we call the *randomized iterate* was introduced by Goldreich, Krawczyk and Luby [GKL93] in the context of constructing pseudorandom generators from regular one-way functions. We revisit this method, simplify existing proofs and utilize our new perspective to achieve significantly better parameters for security and efficiency. We demonstrate that the randomized iterate is also applicable to the construction of pseudorandom generators *from any one-way function*. Specifically we revisit the seminal paper of Håstad, Impagliazzo, Levin and Luby [HILL99] and show that the randomized iterate can help improve the parameters in this content. We also give significant improvements to the construction of pseudorandom generators from one-way functions that are exponentially hard to invert. Finally, we use the randomized iterate both to simplify and to strengthen previous results regarding efficient hardness amplification of regular one-way functions.

We start by introducing the randomized iterate in the context of pseudorandom generators, and postpone the discussion on amplifying weak to strong one-way function to Section 5.1.3.

5.1.1 Pseudorandom Generators and the Randomized Iterate

Pseudorandom Generators, first introduced by Blum and Micali [BM82] and stated in its current, equivalent form, by Yao [Yao82], are one of the cornerstones of cryptography. Informally, a pseudorandom generator is a polynomial-time computable function G that stretches a short random string x into a long string $G(x)$ that "looks" random to any efficient (i.e., polynomial-time) algorithm. Hence, there is no efficient algorithm that can distinguish between $G(x)$ and a truly random string of length $|G(x)|$ with more than a negligible probability. Originally introduced in order to convert a small amount of randomness into a much larger number of effectively random bits, pseudorandom generators have since proved to be valuable components for various cryptographic applications such as bit commitments [Nao91], pseudorandom functions [GGM86], and pseudorandom permutations [LR88], to name a few.

Previous constructions

The first construction of a pseudorandom generator was given in [BM82] based on a particular one-way function and was later generalized in [Yao82] into a construction of a pseudorandom generator based on any one-way permutation. We refer to the resulting construction as the BMY construction. The BMY generator works by iteratively applying the one-way permutation on its own output. More precisely, for a given function f and input x define the k'th iterate recursively as $f^k(x) = f(f^{k-1}(x))$ where $f^1(x) = f(x)$. To complete the construction, one needs to take a hardcore-bit at each iteration. If we denote by $b(x)$ the hardcore-bit of x (take for instance the Goldreich-Levin [GL89] predicate), then the BMY generator on seed x outputs the hardcore-bits $b(f^1(x)), \ldots, b(f^\ell(x))$.[1]

The natural question arising from the BMY generator was whether one-way permutations are actually necessary for pseudorandom generators or can one do with a more relaxed notion. Specifically, is any one-way function sufficient for pseudorandom generators? Levin [Lev87] observed that the BMY construction works for any *one-way function on its iterates*, that is, a one-way function that remains one-way when applied sequentially on its own outputs. A general one-way function, however, does not have this property since the output of f may have very little randomness in it, and a second application of f may be easy to invert. A partial solution was suggested by Goldreich et al. [GKL93] that showed a construction of a pseudorandom generator based on any regular one-way function (referred to as the GKL generator). A regular function is a function such that every element in its image has the same number of preimages. The GKL generator introduced the technique at the core of this work, that we call the ***randomized iterate***. Rather than simple iterations, an extra randomization step is added between every two applications of f. More precisely,

DEFINITION 5.1.1 (the randomized iterate (informal))
For function f, input x and random vector of hash functions $\overline{h} = (h_1, \ldots, h_\ell)$, recursively define the k'th randomized iterate (for $2 \leq k \leq \ell + 1$) by:

$$f^k(x, \overline{h}) = f(h_{k-1}(f^{k-1}(x, \overline{h}))) \ ,$$

where $f^1(x, \overline{h}) = f(x)$.

The rational is that $h_k(f^k(x, \overline{h}))$ is now uniformly distributed, and the challenge is to show that f, when applied to $h_k(f^k(x, \overline{h}))$, is hard to invert even when the randomizing hash functions $\overline{h} = (h_1, \ldots, h_\ell)$ are made public. Once this is shown, the generator is similar in nature to the BMY generator (the generator outputs $b(f^1(x, \overline{h})), \ldots, b(f^\ell(x, \overline{h})), \overline{h}$).

Finally, Håstad et al. [HILL99] (combining [ILL89a, Hås90]), culminated this line of research by showing a construction of a pseudorandom generator using any one-way function (hereafter called the HILL generator). This result is one of the most fundamental and influential theorems in cryptography. It introduced many new ideas that have since proved useful in other contexts, such as the notion of pseudoentropy, and the implicit use of family of pairwise-independent hash functions as randomness extractors. We mention that HILL departs from GKL in its techniques, taking a significantly different approach.

[1] We mention that typically the BMY generator is presented as $b(f^0(x)), \ldots, b(f^{\ell-1}(x))$. However, for consistency with our results, we present it so that the first hardcore bit is taken *after* the first iteration.

The Complexity and Security of the Previous Constructions

While the HILL generator fully answers the question of the plausibility of a generator based on any one-way function, the construction is highly involved and very inefficient. Other than the evident contrast between the simplicity and elegance of the BMY generator to the complex construction and proof of the HILL generator, the parameters achieved in the construction are far worse, rendering the construction impractical.

In practice, it is not necessarily sufficient that a reduction translates polynomial security into polynomial security. In order for reductions to be of any practical use, the concrete overhead introduced by the reduction comes into play. There are various factors involved in determining the security of a reduction, and in Section 2.3.2 we elaborate on the security of cryptographic reductions and the classification of reductions in terms of their security. Here, however, we focus only on one central parameter, which is the length m of the generator's seed compared to the length n of the input to the underlying one-way function. The BMY generator takes a seed of length $m = \Theta(n)$, the GKL generator takes a seed of length $m = \Theta(n^3)$ while the HILL construction produces a generator with seed length on the order of $m = \Theta(n^8)$.[2]

The length of the seed is of great importance to the security of the resulting generator. While it is not the only parameter, it serves as a lower bound to how good the security may be. For instance, the HILL generator on m bits has security that is at best comparable to the security of the underlying one-way function, but only on $\Theta(\sqrt[8]{m})$ bits. To illustrate the implications of this deterioration in security, consider the following example: Suppose that we only trust a one-way function when applied to inputs of at least 100 bits, then the GKL generator can only be trusted when applied to a seed of length of at least one million bits, while the HILL generator can only be trusted on seed lengths of 10^{16} and up (both being highly impractical). Thus, trying to improve the seed length towards a linear one (as it is in the BMY generator) is of great importance in making these constructions practical.

Exponentially hard one-way functions and improving the seed length

The BMY and GKL generators demonstrate that assuming restrictions on the underlying one-way function allows for great improvement of the seed length (or input blowup). The common theme in these restrictions is that they deal with the *structure* of the one-way function. A different approach was recently taken by Holenstein [Hol06a], who builds a pseudorandom generator from any one-way function with *exponential hardness*, i.e., for some constant C, no algorithm of running-time at most 2^{Cn} inverts the function with probability better than 2^{-Cn}. This approach is different as it discusses *raw hardness* as opposed to structure. The result in [Hol06a] is essentially a generalization of the HILL generator that also takes into account the parameter stating the hardness of the one-way function. In its extreme case where the hardness is exponential, the pseudorandom generator takes a seed length of $m = \Theta(n^5)$ and has security $2^{\Theta(m^{\frac{1}{5}})}$. The seed length can be reduced to as low as $\Theta(n^4 \log^2 n)$ when the resulting generator is only required to have super-polynomial security (i.e. security of $n^{\log n}$). In its other extreme based on a general one-way function (with superpolynomial hardness), [Hol06a] forms a formal proof of the best known seed length for the HILL construction (seed length $\Theta(n^8)$).

[2] The seed length actually proved in [HILL99] is $\Theta(n^{10})$, however it is mentioned that a more careful analysis can get to $\Theta(n^8)$. A formal proof for the $\Theta(n^8)$ seed length construction is given by Holenstein [Hol06a].

5.1.2 Our Results on Pseudorandom Generators

Our improvements to the seed length of pseudorandom generators under the various assumptions are summarized in Figure 5.1. In the upcoming section we elaborate on each of these constructions and highlight the source of the improvements.

Paper	Type of function	Seed length
[BM82, Yao82]	One-way permutation	$\Theta(n)$
[GKL93]	Regular one-way function	$\Theta(n^3)$
This work		$\Theta(n \log n)$
[Hol06a]	One-way function with exponential hardness	$\Theta(n^5)$
This work		$\Theta(n^2)$
This work	Regular one-way function with exponential hardness	$\Theta(n)$
[HILL99]	Any one-way function	$\Theta(n^8)$
This work		$\Theta(n^7)$

Figure 5.1: Summary of results.

Regular one-way functions

We give a construction of a pseudorandom generator from any regular one-way function with seed length $\Theta(n \log n)$. We mention that our approach has the potential of reaching a construction with a linear seed, the bottleneck being the efficiency of the currently known bounded-space generators. Our construction follows the randomized iterate method and is achieved in two steps:

- We give a significantly simpler proof that the GKL generator works, allowing the use of a family of hash functions that is pairwise-independent rather than n-wise independent (as used in [GKL93]). This gives a construction with seed length $m = \Theta(n^2)$ (see Theorem 5.3.10).
- The new proof allows for the derandomization of the choice of the randomizing hash functions via the *generator against bounded-space adversaries* (for short, bounded-space generator) of Nisan [Nis92], further reducing the seed length to $m = \Theta(n \log n)$ (see Theorem 5.3.11).

The proof method. Following is a high-level description of our proof method. For simplicity we focus on the second randomized iteration (i.e., on $f^2(x,h) = f(h(f(x)))$, but the same argument generalizes to the other iterations. The main task at hand is to show that it is hard to find $f^1(x,h) = f(x)$ when given $f^2(x,h)$ and h. This follows by showing that any procedure A for finding $f(x)$ given $(f^2(x,h),h)$ enables to invert the one-way function f (on a random image). Specifically, we show that for a random image $z \in f(U_n)$, if we choose a random and independent hash h' and feed the pair (z,h') to A, then A is likely to return a value $f(x')$ such that $h'(f(x')) \in f^{-1}(z)$ (and thus we obtain an inverse of z). This is ultimately shown by proving that if A succeeds on the distribution of $(f^2(x,h),h)$, then A is also successful on the distribution of $(f^2(x,h),h')$ where h' is chosen independently of (x,h).

Our proof is inspired by a technique used by Rackoff in his proof of the Leftover Hash Lemma (in [IZ89]). Rackoff proves that a distribution is close to uniform by showing that it has *collision-*

probability that is very close to that of the uniform distribution.[3] We would like to follow this scheme and consider the collision-probability of the two aforementioned distributions. In our case, however, the two distributions could actually be very far from each other. Yet, with the analysis of the collision-probabilities we manage to prove that the probability of any event under the first distribution is *polynomially related* to the probability of the same event under the second distribution. This proof generalizes nicely also to the case of many iterations.

The derandomization using a bounded-space generator follows directly from the new proof. The point is to introduce a derandomization of the hash functions such that the collision probability of the randomized iterate remains essentially the same. Since the proof centers around the collision probability of $f^\ell(x,\overline{h}),\overline{h}$, the proof will hold also for the derandomized version. More precisely, consider the procedure that given inputs x_0, x_1 and $\overline{h} = (h_1,\ldots,h_{\ell-1})$, outputs one if $f^\ell(x_0,\overline{h})$ equals $f^\ell(x_1,\overline{h})$ and zero otherwise. Note that the probability, over a uniform choice of inputs, that the above procedure outputs one, is exactly the collision-probability of $(f^\ell(x,\overline{h}),\overline{h})$. Also note that the above procedure can run in linear space, since it simply needs to store the two intermediate iterates at each step. Therefore, the probability that the above procedure outputs one while replacing $\overline{h}$ with the output of a generator against linear space adversaries, is very close the collision probability of $(f^\ell(x,\overline{h}),\overline{h})$. It follows that the collision probability of $(f^\ell(x,\widetilde{h}),\widetilde{h})$, where $\widetilde{h}$ is now the output of the bounded-space generator is very close to that of $(f^\ell(x,\overline{h}),\overline{h})$, and the security proof now follows as in the proof when using independent randomizing hash functions. We mention that derandomization of similar spirit was used by Phillips [Phi93], in his efficient amplification of weak one-way permutations (see Section 5.1.3).

Exponentially hard one-way functions

We give a construction of a pseudorandom generator from *any* exponentially-hard one-way function with seed length $m = \Theta(n^2)$ and security $2^{\Theta(\sqrt{m})}$. If we only require the security of the resulting generator to be super-polynomial, then the construction gives seed that is only $\Theta(n\log^2 n)$ long. We mention that Holenstein's result applies for *any* one-way function (but is most efficient when the one-way function is exponentially hard). Our construction on the other hand is specialized for one-way functions with exponential hardness, and does not generalize to use significantly weaker one-way functions. More concretely, our construction can use any one-way function with security $2^{\phi(n)}$, *as long as* $\phi(n) \in \Omega(\frac{n}{\log n})$.

The core technique of our construction is once again the randomized iterate. Trying to apply the randomized iterate to a general one-way function, we encounter the following difficulty: For $k \geq 2$, the k'th randomized iteration of a general one-way function may be easy on a large fraction of the inputs. Our key observation is that the randomized iterate cannot be easy everywhere. Our Lemma 5.4.1 indicates that for every f, there exists a set S^k of inputs to f^k such that the k'th randomized iteration is hard to invert over inputs taken from this set. Moreover, the density of S^k is at least $\frac{1}{k}$. This means that there is some pseudorandomness to be extracted from the k'th randomized iterate: Taking a hard-core bit of the k'th randomized iteration gives a bit that with probability $\frac{1}{k}$ looks random (to a computationally bounded observer). Our idea is to collect these bits that contain some pseudoentropy and to then extract from them the pseudorandom output of the generator.

[3] The collision-probability of a distribution is the probability of getting the same element twice when taking two independent samples from the distribution.

Consider taking m independent copies of the randomized iterate (on m independent inputs) and for each of the m copies taking a hardcore bit from the k'th iteration. This forms a string of m bits, of which $\frac{m}{k}$ are expected to be random looking. Our next step would be to run a *randomness extractor* on this string, to generate $\Theta(\frac{m}{k})$ pseudorandom bits. The problem, however, is that the total number of pseudorandom bits generated, i.e., $\Theta(\sum_{k=1}^{m} \frac{m}{k})$, is too low, and in particular insufficient to compensate for the mn bits invested in the random seed.

This problem can be remedied by taking more hardcore bits at each iteration. Specifically, if the one-way function has exponential hardness then a linear number of hardcore bits may be taken at each iteration (Goldreich and Levin [GL89]). Thus taking $m = n$ independent copies, the total number of pseudorandom bits generated can be larger than the seed length. The construction gives a seed that is $\Theta(n^2)$-long, as each independent copy of the randomized iterate only runs a constant number of iterations.

REMARK 5.1.2 (On randomness extractors and pseudorandomness)
The use of randomness extractors in a computational setting, was initiated in [HILL99]. We give a general "uniform extraction lemma" (Lemma 5.2.2) for this purpose that is proved using a uniform hardcore Lemma of Holenstein from [Hol05]. We mention that a similar proof was given independently in [Hol06a].

Any one-way function

The HILL generator takes a totally different path than the GKL generator. The initial step in the HILL construction takes a one-way function f and generates a bit that has significantly more *pseudoentropy* than actual entropy. This gap is then exploited in order to build a full fledged pseudorandom generator. This initial construction does not use iterations of f at all. We ask whether the technique of randomized-iterations can be helpful for the case of any one-way function, and give a positive answer to this question (actually, we are using only the first two iterations). Specifically, this method also improves the efficiency of the overall construction by an n^3 factor over the original HILL generator and reduces the seed length by a factor of n (which also implies improvement in the security of the construction). All in all, we present a pseudorandom generator from *any* one-way function with seed length $\Theta(n^7)$ (Theorem 5.5.10) which is the best known to date.

Our generator replaces the initial step of the HILL generator with a different construction based on the techniques we have developed. We briefly describe the new initial step. Denote the degeneracy of y by $\mathrm{Deg}_f(y) = \lceil \log |f^{-1}(y)| \rceil$ (this is a measure that divides the images of f to n categories according to their preimage size). Let b denote a hardcore-bit (again we take the Goldreich-Levin hardcore-bit [GL89]). Loosely speaking, we consider the bit $b(f(x))$ when given the value $(f^2(x,h),h)$ and make the following observation. When $\mathrm{Deg}_f(f(x)) \geq \mathrm{Deg}_f(f^2(x,h))$ the value $b(f(x))$ is (almost) fully determined by $(f^2(x,h),h)$, as opposed to when $\mathrm{Deg}_f(f(x)) < \mathrm{Deg}_f(f^2(x,h))$ where no information about $b(f(x))$ leaks. But in addition, if $\mathrm{Deg}_f(f(x)) = \mathrm{Deg}_f(f^2(x,h))$, then $b(f(x))$ is computationally-indistinguishable from uniform (that is, looks uniform to any efficient observer), even though it is actually fully determined. The latter stems from the fact that when $\mathrm{Deg}_f(f(x)) = \mathrm{Deg}_f(f^2(x,h))$ the behavior is close to that of a regular function.

As a corollary we get that the bit $b(f(x))$ has entropy of no more than $\frac{1}{2}$ (i.e., the proba-

bility of $\mathrm{Deg}_f(f(x)) < \mathrm{Deg}_f(f^2(x,h)))$, but has "entropy of at least $\frac{1}{2} + \frac{1}{\Theta(n)}$ in the eyes of any computationally-bounded observer" (i.e., the probability of $\mathrm{Deg}_f(f(x)) \leq \mathrm{Deg}_f(f^2(x,h)))$. In other words, $b(f(x))$ has entropy $\frac{1}{2}$ but pseudoentropy of $\frac{1}{2} + \frac{1}{\Theta(n)}$.[4] As in HILL, it is this gap of $\frac{1}{\Theta(n)}$ between the entropy and pseudoentropy that eventually allows the construction of a pseudorandom generator.

Comparing to HILL. The HILL construction builds a pair of function and predicate such that the predicate has entropy p, but pseudoentropy of at least $p + \frac{1}{\Theta(n)}$ (see Appendix 5.6 for the description of their pair). Unlike in our construction, however, the entropy threshold p in the HILL construction is unknown (i.e., not efficiently computable). This is a real disadvantage, since knowledge of this threshold is essential for the overall construction. To overcome this, the HILL generator enumerates all values for p (up to an accuracy of $\Theta(\frac{1}{n})$), runs the generator with each of these values and eventually combines all generators using an XOR of their outputs. This enumeration costs an additional factor n to the seed length as well an additional factor of n^3 to the number of calls to the underlying function f, and hence our efficiency and security improvements.

REMARK 5.1.3 (On pseudorandomness in NC^1)
For the most part, the HILL construction is "depth" preserving. In particular, given two "non-uniform" hints of $\log n$ bits each (that specify two different properties of the one-way function [5]), the reduction gives generators in NC^1 from any one-way function in NC^1. Unfortunately, without these hints, the depth of the construction is polynomial (rather than logarithmic). Our construction eliminates the need for one of these hints (we still need to know the entropy of the function) and thus can be viewed as a step towards achieving generators in NC^1 from any one-way function in NC^1. Building pseudorandom generators in NC^1 (from one-way functions in NC^1) would be highly significant. In particular, since [AIK06] showed that such generators imply pseudorandom generators in NC^0.

5.1.3 One-way Functions - Amplification from Weak to Strong

The existence of one-way functions is essential to almost any task in cryptography (see for example [IL89]) and also sufficient for numerous cryptographic primitives such as the pseudorandom generators discussed above. In general, for constructions based on one-way functions we use what are called *strong* one-way functions. That is, functions that can only be inverted efficiently with negligible success probability. A more relaxed definition is that of a δ-weak one-way function where $\delta(n)$ is a polynomial fraction. This is a function that every efficient algorithm fails to invert on at least a $\delta(n)$ fraction of the inputs. This definition is significantly weaker, yet, Yao [Yao82] showed how to convert any weak one-way function into a strong one (see proof in [Gol01a]). The new strong one-way function simply consists of many independent copies of the weak function concatenated to

[4] It natural to ask why should we consider $b(f(x))$ as the predicate and not simply $b(x)$. Clearly the pseudoentropy of $b(x)$, given $(f^2(x,h),h)$ is at least as large as that of $b(f(x))$ (since $f(x)$ is determined by x). The problem is that the real entropy of $b(x)$ in this case is unknown (and may, in fact, be as high as the pseudoentropy). In other worlds, when considering $b(f(x))$ rather than $b(x)$, we reduce the conditional entropy to a known bound, while keeping the pseudoentropy larger than this bound.

[5] Consider the random variable induced by applying the one-way function on a uniformly chosen input. One of these hints relates to the entropy of this random variable, where the other hint, p, relates to its variance.

each other. The solution of Yao, however, incurs a blow-up factor of $\omega(\log(n)/\delta(n))$ to the input length of the strong function, which translates to a significant loss in the security (as in the case of pseudorandom generators).

Goldreich et al. [GIL+90] pointed out this loss of security problem and gave a solution for one-way permutations that has just a linear blowup in the length of the input. Their solution was also generalized to *known-regular* one-way functions (regular functions whose image size is efficiently computable), where its input length varied according to the required security. The input length is linear when the required security is $2^{O(\sqrt{n})}$, but deteriorates up to $\Theta(n^2)$ when the required security is higher (e.g., security $2^{\Theta(n)}$).[6] Their construction uses a variant of randomized iterates where the randomization is via one random step on an expander graph. Additional attempts to avoid this loss of security problem were given by [Phi93, DI99] (see below).

Our Contribution to Hardness Amplification

We present an alternative efficient hardness amplification for regular one-way functions. Specifically, in Theorem 6.2.1 we show that the m'th randomized iterate of a weak one-way function along with the randomizing hash functions form a strong one-way function (for the right choice of m). Moreover, this holds also for the derandomized version of the randomized iterate (Theorem 6.2.8), giving an almost linear construction. Our construction is arguably simpler and has the following advantages:

1. While the [GIL+90] construction works only for *known* regular weak one-way functions, our amplification works for any regular weak one-way function (whether its image size is efficiently computable or not).

2. The input length of the resulting strong one-way function is $\Theta(n \log n)$ regardless of the required security. Thus, for some range of the parameters our solution is better than that of [GIL+90] (although it is worse than [GIL+90] for other ranges).

We mention that our method may yield a construction with input length $\Theta(n)$ if bounded-space generators with better parameters become available.

The Idea. At the basis of all hardness amplification lies the fact that for any inverting algorithm, a weak one-way function has a set that the algorithm fails upon, hereafter called the ***failing-set*** of this algorithm. The idea is that a large enough number of randomly chosen inputs are bound to hit every such failing-set and thus to fail every algorithm. Taking independent random samples (i.e., $f'(x_1,\ldots,x_m) = (f(x_1),\ldots,f(x_m))$) works well ([Yao82]), but with the price of increasing the input length. An alternative approach (a variant of which was used by [GIL+90]) would be to use randomized iterations (i.e., to consider the function $f^m(x,\overline{h})$). This also amounts to applying f to m random inputs and therefore bound to hit every failing set. At a first glance, this approach does not help in decreasing the input blowup, since the description of $\overline{h}$ is long. Yet, this approach turns out to be beneficial as will be described shortly. Another obstacle is that when given $f^m(x,\overline{h})$, an adversary may invert f^m to a different input $(x',\overline{h}')$ (with different and carefully chosen hash functions) such that the computation of $f^m(x',\overline{h}')$ avoids applying f to a relevant failing set. To

[6] Loosely speaking, one can think of the security as the probability of finding an inverse to a random image $f(x)$ simply by choosing a random element in the domain.

overcome this, the hash functions $\overline{h}$ are also given as part of the output (i.e., we consider the function $g(x,\overline{h}) = (f^m(x,\overline{h}),\overline{h})$), hence forcing an inverter to invert to the same hash functions. Using our core technique (from the pseudorandom generators section) we show that the hardness of inverting f^m is maintained even when $\overline{h}$ is known.

As mentioned above, a basic problem with this approach is that choosing fully independent randomizing hash functions requires an input as long as that of Yao's solution (an input of length $\Theta(n \cdot \omega(\log(n))/\delta(n))$). What makes this approach appealing, is the derandomization of the hash functions using space-bounded generators, which reduces the input length to only $\Theta(n \log n)$. We mention that since the hardness of f^m stems from the fact that a random input hits with high probability any failing-set, it is required that this is also the case for the derandomized f^m (and not only that the derandomized function maintains low collision-probability as in the pseudorandom generator case). Fortunately, the derandomization using bounded-space generators also guarantees this property.

We mention that there have been several attempts to formulate such a construction, using all of the aforementioned tools. Goldreich et al. [GIL+90] did actually consider following the GKL methodology, but chose a different (though related) approach. Phillips [Phi93] gives a solution with input length $\Theta(n \log n)$ using bounded-space generators, but only for the simple case of permutations (where [GIL+90] has better parameters). Di Crescenzo and Impagliazzo [DI99] give a solution for regular functions, but only in a model where public randomness is available (in the mold of [HL92]). Their solution is based on pairwise-independent hash functions that serve as the public randomness. We are able to combine all these ingredients into one general result, perhaps due to our simplified proof.

Additional Issues

On non-length-preserving functions. This work focuses on length-preserving one-way functions. We also demonstrate how our proofs may be generalized, with no penalty in the tightness of the security, to use non-length preserving functions.[7] This generalization requires the use of a construction of a family of *almost* pairwise-independent hash functions (see Sections 2.3.1 and 5.3.4).

The results in the public randomness model. Similarly to previous works, our results also give linear reductions in the public randomness model. This model (introduced by Herzberg and Luby [HL92]) allows the use of public random coins that are not regarded a part of the input. Our results, however, introduce significant savings in the amount of public randomness that is necessary.

5.1.4 Outline

Section 5.2 includes the additional definitions and notations used throughout this chapter. In Section 5.3 we present our construction of pseudorandom generators from regular one-way functions. Section 5.4 presents the construction based on exponentially hard one-way functions and in particular proves a lemma regarding the hardness of inverting the randomized iterate of a general

[7] Previously used techniques for converting arbitrary one-way functions to length-preserving ones (e.g., padding), incur serious deterioration in the security of the resulting one-way functions.

one-way function (Lemma 5.4.1). Finally, in Section 5.5 we present our improvement to the HILL construction of pseudorandom generators from any one-way function.

5.2 Preliminaries

5.2.1 Hardcore Predicates and Functions

Hard-core predicates/functions have a major role in the construction of pseudorandom generators based on one-way functions.

DEFINITION 5.2.1 (hardcore functions)
Let t and δ be functions from $\mathbb{N}$ to $\mathbb{N}$ and from $\mathbb{N}$ to $[0,1]$ respectively, let $L_n \subseteq \{0,1\}^n$ and let $f : \{0,1\}^n \mapsto \{0,1\}^*$ and $hc : \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}$ be two functions defined over $\{0,1\}^n$. We call hc a $(t(n), \delta(n))$-hardcore function of f over L_n, if it is polynomial-time computable and for every algorithm A with running time at most $t(n)$

$$\Delta^A\left((f(x), hc(x))_{x \leftarrow L_n}, (f(x), U_{\ell(n)})_{x \leftarrow L_n}\right) < \delta(n) \ ,$$

for all but finitely many n's.

In the case that $\delta(n) = \frac{1}{t(n)}$, we simply say that hc is a $t(n)$-hardcore function. If hc is $p(n)$-hardcore function for every polynomial p, then we call it an hardcore function of f. As a convention in the case that $L_n = \{0,1\}^n$, it is omitted it form the definition of hc. If hc is a predicate (i.e., $\ell(n) = 1$), then it is called an hardcore predicate of f. Finally, it is custom to call the value $hc(x)$, the "hardcore-bits" of $f(x)$.

The following theorem is an immediate generalization of the Goldreich-Levin hardcore function [GL89, Corollary 1].

THEOREM 5.2.2
There exists an efficiently computable family of functions $\left\{gl_i : \{0,1\}^{3n} \mapsto \{0,1\}^i\right\}_{i \in [n]}$, a constant $c \in (0,1)$ and a polynomial p such that the following holds. Let $L_n \subseteq \{0,1\}^n$, let f and g be two polynomial-time computable functions from $\{0,1\}^n$ to $\{0,1\}^n$ and assuming that for every algorithm A with running time at most $t(n)$ it holds that

$$\Pr_{x \leftarrow L_n}[A(f(x)) = g(x)] \leq \frac{1}{t(n)} \ .$$

Then the following holds for every function t' satisfying $p(n, t'(n)) < t(n)$. For every value of $\ell \in \{1, \ldots, \lfloor c \log(t(n)) \rfloor\}$, the function $hc : \{0,1\}^{3n} \mapsto \{0,1\}^\ell$ defined as $hc(x,r) = gl_\ell(g(x), r)$, is a $t'(n)$-hardcore function of $f'(x,r) = (f(x), r)$ over L_n.

5.2.2 A Uniform Extraction Lemma

The following lemma is a generalization of the (uniform version) of Yao's XOR lemma. Given m independent $(t, 1-\delta)$-hardcore bits, we would like to extract approximately δm pseudorandom bits out of them. The version we present here generalizes [HILL99, Lemma 6.5]), in particular the

original lemma required the hardcore predicate to have a hardcore-set (i.e., a subset of inputs such that the value of the predicate is unpredictable [computationally] over this subset), where in the following lemma this property is no longer required. In addition, the original lemma was tailored for the specific function and predicate it was used with, where the following lemma suits any hard predicate. Finally, the original lemma is stated using an efficient family of pairwise-independent hash functions, while the following lemma is stated using any explicit randomness extractor. The lemma is proven using Holenstein's "uniform hardcore lemma" [Hol05].[8]

LEMMA 5.2.3

Let $f : \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}$ be a polynomial-time computable function, let b be a $(t(n), 1-\delta(n))$-hardcore predicate of f and let $\mathrm{Ext} : \{0,1\}^m \times \{0,1\}^d \mapsto \{0,1\}^r$ be a (k,ε)-strong-extractor. We define $hc : \{0,1\}^{mn} \times \{0,1\}^d \mapsto \{0,1\}^r$ as $hc(x_1,\ldots,x_m,y) = \mathrm{Ext}(y, b(x_1),\ldots,b(x_m))$ and let $f'(x_1,\ldots,x_m,y) = (f(x_1),\ldots,f(x_m),y)$.

There exists a polynomial p such that the following holds. For any $\gamma(n) > m \cdot 2^{-n/4}$ and $t'(n)$ satisfying $p(t'(n), 1/\gamma(n), 1/\delta(n), m, n) < t(n)$, the function hc is a $(t'(n), \varepsilon + \rho(n) + \gamma(n))$-hardcore function of f', where $\rho(n)$ is the probability that when taking m independent samples in $\{0,1\}^n$ less than k samples are in some fixed subset of density $\delta(n)$.

Proof. For clarity we omit the value n whenever it is clear from the context. Let A be an algorithm that runs in time t_A and given $f'(x_1,\ldots,x_m,y)$, distinguishes the value of $hc(x_1,\ldots,x_m,y)$ from random with advantage $\varepsilon_A > \varepsilon+\rho+\gamma$. For a fixed set $S \subseteq \{0,1\}^n$ of density δ, define the following, not necessarily efficiently computable, randomized predicate $Q : \{0,1\}^n \mapsto \{0,1\}$.

$$Q(x) = \begin{cases} U_1 & x \in S, \\ b(x) & \text{otherwise.} \end{cases} \tag{5.1}$$

For any $i \in \{0,\ldots,m\}$ we define the distribution D^i as:

$$D^i = (f(U_n^1),\ldots,f(U_n^1), U_d, \mathrm{Ext}(U_d, b(U_n^1),\ldots,b(U_n^i), Q(U_n^{i+1}),\ldots,Q(U_n^m)).$$

We first note that D^m is equal to $(f'(U_n^1,\ldots,U_n^m,U_d), hc(U_n^1,\ldots,U_n^m,U_d))$ (i.e., to the output of f' concatenated with our candidate hardcore function hc). Also, since with probability $1-\rho$ the min-entropy of D^0 is k, then by the properties of the extractor we have that the statistical distance between D^0 and $(f(U_n^1),\ldots,f(U_n^1),U_d,U_r)$ is at most $\varepsilon+\rho$. It follows that A distinguishes between D^0 and D^m with advantage $\varepsilon_A - \rho - \epsilon > \gamma$. Hence, by a standard hybrid argument we deduce that there exists a $j \in \{0,\ldots,m-1\}$ such that A distinguishes between D^j and D^{j+1} with probability γ/m. Moreover, Since D^j and D^{j+1} are identical when $x_{j+1} \notin S$, it follows that A must achieve this distinguishing probability between D^j and D^{j+1} also when given that $x_{j+1} \in S$. Finally, note that the only difference between D^j and D^{j+1} given that $x_{j+1} \in S$ is whether the $(j+1)$'th input to Ext is $b(x_j)$ or a random input. The following algorithm given oracle access to the characteristic function χ^S of any set of density δ, returns, with probability $1-2^{-n}$, a circuit that distinguishes between $(f(x),b(x))_{x\leftarrow S}$ and $(f(x),U)_{x\leftarrow S}$ with probability γ/m.

[8] Independently of this work, [Hol06b, Theorem 7.3] presents a different version of Lemma 5.2.3. While their theorem yields some generalization over the following lemma (informally it also considers the situation where some information about the hardcore bits leaks to the adversary), it is only stated for the case of polynomial hardness.

ALGORITHM 5.2.4
Algorithm B^{χ^S}.

1. Find a $j \in \{0,\ldots,m-1\}$ such that, with probability at least $1-2^{-n}$, A distinguishes between D^j and D^{j+1} with success probability at least γ/m. This can be done using χ^S, by sampling form distributions D^j and D^{j+1} and running A on them.[9]
2. Sample d^j, an instance of D^j (again using χ^S).
3. Return the circuit C that on input (y,b) replaces $f(x_{j+1})$ and $b(x_{j+1})$ in d^j by y and b respectively and outputs $A(d^j)$.

Note that t_B, the running-time of B, is polynomial in t_A, δ, m, δ and n. For any fixed $S \subseteq \{0,1\}^n$ of density δ let $adv_S = \mathsf{Ex}[C \leftarrow B^{\chi^S} : \Delta^C((X_n, b(X_n)),(f(X_n),U))]$, where X_n is uniformly distributed over S_n and the expectation is over the random coins of B. By the above observations it follows that $adv_S > \gamma - 2^{-n} > 2^{-n/3}$. By [Hol06a, Propostition 3] there exists an algorithm that runs in time polynomial in n, m, t_B and $1/adv_S$, and distinguishes between $(f(U_n), b(U_n))$ and $(f(U_n), U)$ with probability at least $1-\delta$.[10] Thus, choosing p to accommodate the running time of B and of the algorithm guaranteed by [Hol06a, Propostition 3], the hardness of b yields that $p(t_A, 1/\gamma, 1/\delta, m, n) \geq t(n)$. □

5.2.3 Pseudorandom Generators

DEFINITION 5.2.5 (pseudorandom generators)
Let t and δ be functions from $\mathbb{N}$ to $\mathbb{N}$ and to $[0,1]$ respectively, and let $G : \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}$ be a polynomial-time computable function where $\ell(n) > n$. We say that G is a $(t(n),\delta(n))$-pseudorandom-generator if for every algorithm A with running time at most $t(n)$

$$\Delta^A\left(G(U_n), U_{\ell(n)}\right) < \delta(n) \ .$$

In the case that $\delta(n) = \frac{1}{t(n)}$, we simply say that G is a $t(n)$-pseudorandom-generator. G is a pseudorandom-generator if it is $p(n)$-pseudorandom-generator for every polynomial p.

5.2.4 Bounded-Space Generators

Bounded-Space generators are pseudorandom generators against bounded-space adversaries. Such generators plays a central role in derandomization tasks. We are interested in generator for the following type of adversaries.

DEFINITION 5.2.6 (bounded-width layered branching program - LBP)
A (s,t,v)-LBP M is a directed graph with $2^s \cdot (t+1)$ vertices, partitioned into $t+1$ layers with 2^s vertices in each layer. Each vertex in the i'th layer has exactly 2^v outgoing labeled edges to the

[9] Alternatively, one can just pick a random $j \in \{0,\ldots,m-1\}$ and succeed with probability at least $\frac{1}{m}$.
[10] The original form of the uniform hardcore lemma, [Hol05, Lemma 2.5], was stated only for super polynomial hardness and [Hol06a, Propostition 3] generalizes it for any hardness.

$(i+1)^{st}$ layer, one for every possible string $z \in \{0,1\}^v$. The vertices in the last layer (the t layer) are labeled by 0 or 1.

Denote by M_x such a LBP with starting vertex $x \in \{1,\ldots,2^s\}$ in the 0'th level. For a sequence $\overline{z} \in \{0,1\}^{tv}$, we define the output of the LBP $M_x(\overline{z})$ by a walk on the graph starting at vertex x in layer 0 and advancing to the i'th layer along the edge labeled by $\overline{z}_i$. $M_x(\overline{z})$ accepts if it reaches a vertex labeled by 1 and rejects otherwise.

DEFINITION 5.2.7
A generator $G : \{0,1\}^m \mapsto \{0,1\}^{tv}$ is said to ε-fool a LBP M if for every $x \in \{1,\ldots,2^s\}$ we have:

$$|[M_x(U_{tv}) \text{ accepts}] - \Pr[M_x(G(U_m)) \text{ accepts}]| < \varepsilon$$

THEOREM 5.2.8 [Nis92, INW94]
For every s,t,v there exist a generator $\text{BSG} : \{0,1\}^{\Theta(v+(s+\log(\frac{t}{\varepsilon}))\log t)} \mapsto \{0,1\}^{tv}$ running in time $\text{poly}(s,t,v)$ that ε-fools every (s,t,v)-LBP.

5.3 Pseudorandom Generators from Regular One-way Functions

The following discussion considers only length preserving regular one-way functions. Note that by Section 5.3.4, this is without lost of generality.

5.3.1 Some Motivation and the Randomized Iterate

Recall that the BMY generator simply iterates the one-way permutation f on itself, and outputs a hardcore-bit of the intermediate step at each iteration. The crucial point is that the output of the function is also uniform in $\{0,1\}^n$ since f is a permutation. Hence, when applying f to the output, it is hard to invert this last application of f, and therefore hard to predict the new hardcore-bit (Yao shows [Yao82] that the unpredictability of bits implies pseudorandomness). Since the seed is essentially just an n bit string and the output is as long as the number of iterations, then the generator actually stretches the seed.

We want to duplicate this approach for general one-way functions, unfortunately the situation changes drastically when the function f is not a permutation. After a single application of f, the output may be very far from uniform, and in fact may be concentrated on a very small and easy fraction of the inputs to f. Thus reapplying f to this output gives no hardness guarantees at all. In an attempt to salvage the BMY framework, Goldreich et. al. [GKL93] suggested to add a randomization step between every two applications of f, thus making the next input to f a truly random one. This modification that we call randomized iterates lies at the core of our work and is defined next.

DEFINITION 5.3.1 (the randomized iterate)
Let $f : \{0,1\}^n \mapsto \{0,1\}^n$, let $\mathcal{H}$ be an efficient family of pairwise-independent length-preserving hash functions defined over $\{0,1\}^n$ and let $\ell \in \mathbb{N}$. For $2 \leq k \leq \ell+1$, $x \in \{0,1\}^n$ and $\overline{h} \in \mathcal{H}^\ell$ define the k'th randomized iterate $f^k : \{0,1\}^n \times \mathcal{H}^\ell \mapsto \text{Im}(f)$ recursively as

$$f^k(x,\overline{h}) = f(\overline{h}_{k-1}(f^{k-1}(x,\overline{h}))) \ ,$$

where $f^1(x,\overline{h}) = f(x)$. In the following we denote by H^k the random variable uniformly distributed over $\mathcal{H}^k$.

The application of the randomized iterate for pseudorandom generators is a bit tricky. On the one hand, such a randomization costs a large number of random bits, much larger than what can be compensated for by the hardcore-bits generated in each iteration. So in order for the output to actually be longer than the input, we also output the descriptions of the hash functions. But on the other hand, handing out the randomizing hash gives information on intermediate values such as $f^i(x,\overline{h})$, and f might no longer be hard to invert when applied to such an input. Somewhat surprisingly, the randomized iterate of a regular one-way function remains hard to invert even when the hash functions are known. This fact, which is central to the whole approach, was proved in [GKL93] when using a family of n-wise independent hash functions. As a first step, we give a simpler proof that extends to pairwise-independent hash functions as well.

5.3.2 The Last Randomized Iteration is Hard to Invert

In this section we formally state and prove the key observation mentioned above. After applying k randomized-iterations of a regular one-way function f, it is hard to invert the last-iteration, even if given access to all of the hash functions leading up to this point.

LEMMA 5.3.2
Let $f : \{0,1\}^n \mapsto \{0,1\}^n$ be a regular $(t(n),\delta(n))$-one-way, let $k \in \text{poly}(n)$, and let f^k and $\mathcal{H}$ be as in Definition 5.3.1. Then there exists a polynomial p, such that for every algorithm A of running-time at most $(t(n) - p(n))$ it holds that

$$\Pr[A(f^k(U_n, H^{k-1}), H^{k-1}) = f^{k-1}(U_n, H^{k-1})] \leq \sqrt[3]{8k\delta(n)}$$

for large enough n, where the probability is also taken over the random coins of A.

We briefly give some intuition to the proof, illustrated with regard to the first randomized iterate. Suppose that we have an algorithm A that *always* finds $f^1(x,h) = f(x)$ given $f^2(x,h)$ and h. In order to invert the one-way function f on an element $y \in \text{Im}(f)$, we simply need to find a hash h' that is consistent with y, in the sense that there exists an x' such that $y = f^2(x',h')$. Now we simply run $z = A(y,h')$, and output $h'(z)$ (and indeed $f(h'(z)) = y$). The point is that if f is a regular function, then finding a consistent hash is easy, simply because a random and independent h' is likely to be consistent with y. The actual proof follows this framework, but is far more involved due to the fact that the reduction starts with an algorithm A that has only a small (yet polynomial) success probability.

Proof. Let A be an algorithm with running-time $t_A(n)$ such that

$$\Pr[A(f^k(U_n, H^{k-1}), H^{k-1})) = f^{k-1}(U_n, H^{k-1})] \geq \varepsilon_A(n) \ ,$$

where $\varepsilon_A(n) > \sqrt[3]{8k\delta(n)}$. We show that unless t_A is large, we can use A in order to violate the hardness of f. Consider the procedure M^A for this task.

ALGORITHM 5.3.3

Algorithm M^A for inverting f.
Input: $y \in \mathrm{Im}(f)$.

1. Choose uniformly at random $\overline{h} \in \mathcal{H}^{k-1}$.
2. Apply $A(y, \overline{h})$ to get an output x.
3. Output $\overline{h}_{k-1}(x)$.

Letting $p(n)$ be the sampling and evaluation time of $\overline{h}$, we have that the running-time of M^A is at most $t(n)$. The rest of the proof of Lemma 5.3.2 shows that M^A succeeds with probability at least $\varepsilon_A(n)^3/8k > \delta(n)$ on uniformly chosen $y \in \mathrm{Im}(f)$, and we conclude that $t_A(n) > t(n) - p(n)$.

We start by focusing our attention only on those inputs for which A succeeds reasonably well. Recall that the success probability of A is taken over the choice of inputs to A as induced by the choice of $x \in \{0,1\}^n$ and $\overline{h} \in \mathcal{H}^{k-1}$ and the internal coin-tosses of A. The following Markov argument implies that the probability of getting an element in the set that A succeeds on is not very small.

CLAIM 5.3.4

Let $S_A \subseteq \mathrm{Im}(f^k) \times \mathcal{H}^{k-1}$ be the subset defined as

$$S_A = \left\{ (y, \overline{h}) \in \mathrm{Im}(f^k) \times \mathcal{H}^{k-1} : \Pr[f(\overline{h}_{k-1}(A(y, \overline{h})) = y] > \varepsilon_A(n)/2 \right\} \ ,$$

then

$$\Pr[(f^k(U_n, H^{k-1}), H^{k-1}) \in S_A] \geq \varepsilon_A(n)/2 \ .$$

Proof. Suppose that $\Pr[(f^k(U_n, H^{k-1}), H^{k-1}) \in S_A] < \varepsilon_A(n)/2$. Then we have:

$$\begin{aligned} & \Pr[A(f^k(U_n, H^{k-1}), H^{k-1}) = f^{k-1}(U_n, H^{k-1})] \\ < \ & \frac{\varepsilon_A(n)}{2} \cdot \Pr[(f^k(U_n, H^{k-1}), H^{k-1}) \notin S_A] + 1 \cdot \Pr[(f^k(U_n, H^{k-1}), H^{k-1}) \in S_A] \\ < \ & \frac{\varepsilon_A(n)}{2} + \frac{\varepsilon_A(n)}{2} = \varepsilon_A(n) \ . \end{aligned}$$

which contradicts the assumption about the success probability of A. □

Now that we identified a heavy subset of the inputs that A succeeds upon, we want to say that M^A has a fair chance to hit outputs induced by this subset. This is formally shown in the following lemma.

LEMMA 5.3.5

For every set $T \subseteq \mathrm{Im}(f^k) \times \mathcal{H}^{k-1}$, if

$$\Pr[(f^k(U_n, H^{k-1}), H^{k-1}) \in T] \geq \gamma \ ,$$

then

$$\Pr[(f(U_n), H^{k-1}) \in T] \geq \frac{\gamma^2}{k} \ .$$

Assuming Lemma 5.3.5 we may conclude the proof of Lemma 5.3.2. Claim 5.3.4 yields that $\Pr[(f^k(U_n, H^{k-1}), H^{k-1}) \in S_A] \geq \frac{\varepsilon_A(n)}{2}$. By Lemma 5.3.5 taking $T = S_A$ and $\gamma = \varepsilon_A(n)/2$ we get that $\Pr[(f(U_n), H^{k-1}) \in S_A] \geq \varepsilon_A(n)^2/4k$. Thus M^A has a $\varepsilon_A(n)^2/4k$ chance of hitting the set S_A on which it will succeed with probability at least $\varepsilon_A(n)/2$. Altogether, M^A succeeds in inverting f with probability $\varepsilon_A(n)^3/8k > \delta(n)$. □

Proof. (of Lemma 5.3.5) The lemma essentially states that with respect to $\widetilde{f^k}(x, \overline{h}) = (f^k(x, \overline{h}), \overline{h})$ (i.e, the function defined by concatenating the input hash functions to the output of f^k), any large subset of inputs induces a large subset of outputs. Thus, there is a fairly high probability of hitting this output set simply by sampling independent y and $\overline{h}$. Intuitively, if a large set of inputs induces a small set of outputs, then there must be many collisions in this set (a collision means that two different inputs lead to the same output). We show that this is impossible, however, by proving that the collision-probability of the function $(f^k(x, \overline{h}), \overline{h})$ is small.

CLAIM 5.3.6

$$\mathrm{CP}((f^k(U_n, H^{k-1}), H^{k-1}) \leq \frac{k}{|\mathcal{H}|^{k-1} \cdot |\mathrm{Im}(f)|}$$

Proof. For every two inputs $(x_0, \overline{h_0})$ and $(x_1, \overline{h_1})$ to f^k, in order to have a collision we must first have that $\overline{h_0} = \overline{h_1}$, which happens with probability $(1/|\mathcal{H}|)^{k-1}$. Now, given that $\overline{h_0} = \overline{h_1} = \overline{h}$ (with a random $\overline{h} \in \mathcal{H}^{k-1}$), we require also that $f^k(x_0, \overline{h})$ equals $f^k(x_1, \overline{h})$. If $f(x_0) = f(x_1)$ (happens with probability $1/|\mathrm{Im}(f)|$) then a collision is assured. Otherwise, there must be an $i \in [k-1]$ for which $f^i(x_0, \overline{h}) \neq f^i(x_1, \overline{h})$ but $f^{i+1}(x_0, \overline{h}) = f^{i+1}(x_1, \overline{h})$. Since $f^i(x_0, \overline{h}) \neq f^i(x_1, \overline{h})$, due to the pairwise-independence of h_i, the values $h_i(f^i(x_0, \overline{h}))$ and $h_i(f^i(x_1, \overline{h}))$ are uniformly random values in $\{0,1\}^n$, and thus $f(h_i(f^i(x_0, \overline{h}))) = f(h_i(f^i(x_1, \overline{h})))$ happens with probability $1/|\mathrm{Im}(f)|$. Altogether, $\mathrm{CP}((f^k(U_n, \mathcal{H}^{k-1}), \mathcal{H}^{k-1}) \leq \frac{1}{|\mathcal{H}|^{k-1}} \cdot \frac{k}{|\mathrm{Im}(f)|}$. □

On the other hand, we check the probability of getting a collision inside the set T, which is a lower bound on the probability of getting a collision at all. We first request that both $(x_0, \overline{h_0}) \in T$ and $(x_1, \overline{h_1}) \in T$. This happens with probability at least γ^2. Then, once inside T, we know that the probability of collision is at least $1/|T|$. Altogether:

$$\mathrm{CP}(f^k(U_n, H^{k-1}), H^{k-1}) \geq \gamma^2 \cdot \frac{1}{|T|} \quad . \tag{5.2}$$

Combining Claim 5.3.6 and (5.2), we get $\frac{|T|}{|\mathcal{H}|^{k-1} \cdot |\mathrm{Im}(f)|} \geq \frac{\gamma^2}{k}$. Since the probability of getting a value in T when choosing a random element in $\mathrm{Im}(f) \times \mathcal{H}^{k-1}$ is exactly $\frac{|T|}{|\mathcal{H}|^{k-1}|\mathrm{Im}(f)|}$ (this follows since for a regular function f the distribution $f(U_n)$ is the uniform distribution over $\mathrm{Im}(f)$). It follows that $\Pr[(y, \overline{h}) \in T] \geq \gamma^2/k$ as requested. □

REMARK 5.3.7

The last fact in the proof of Lemma 5.3.5 (that $f(U_n)$ is uniformly distributed over $\mathrm{Im}(f)$) is in

fact the only place where the regularity of the one-way function is required. The proof fails for general one-way functions where the preimage size of different elements may vary considerably. For a general function, the collision-probability argument at the heart of this lemma can be made to hold only as long as the last element in a sequence of applications is at least as heavy as all the elements along the sequence (see Lemma 5.4.1). While for regular functions this requirement is always true, for general one-way functions this occurs with probability that deteriorates linearly (in the length of the sequence). Thus using a long sequence of iterations is likely to lose the hardness of the original one-way function (see more in Section 5.4).

5.3.3 A Pseudorandom Generator from a Regular One-way Function

After showing that the randomized-iterations of a regular one-way function are hard to invert, it is natural to follow the footsteps of the BMY construction to construct a pseudorandom generator. Rather than using simple iterations of the function f, randomized-iterations of f are used instead, with fresh randomness in each application. As in the BMY case, a hardcore-bit(s) of the current input is taken at each stage. In order to keep things more readable, we start by giving our pseudorandom generator based on regular one-way functions with super polynomial hardness (i.e., standard one-way functions). In Section 5.3.3 generalize this result to regular one-way functions with arbitrary hardness. In particular, we get more efficient pseudorandom generators, assuming that underlying regular one-way functions are exponentially hard.

THEOREM 5.3.8

Let $f : \{0,1\}^n \mapsto \{0,1\}^n$ be a regular one-way function and let $\mathcal{H}$ be an efficient family of pairwise-independent length preserving hash functions. Define $G : \{0,1\}^n \times \mathcal{H}^n \times \{0,1\}^{2n} \mapsto \{0,1\}^{n+1} \times \mathcal{H}^n \times \{0,1\}^{2n}$ as

$$G(x, \overline{h}, r) = (b(f^1(x, \overline{h}), r), \ldots, b(f^{n+1}(x, \overline{h}), r), \overline{h}, r),$$

where:

- b is the Goldreich-Levin hardcore predicate (see Theorem 5.2.2),
- Recall that $f^1(x, \overline{h}) = f(x)$ and for $2 \leq i \leq n+1$ it holds that $f^i(x, \overline{h}) = f(\overline{h}_{i-1}(f^{i-1}(x, \overline{h})))$.

Then, G is a pseudorandom generator.

Proof. Lemma 5.3.2 yields that no efficient algorithm can compute f^{k-1} given $(f^k, h_1, \ldots, h_{k-1})$ with non-negligible success probability. Hence, Theorem 5.2.2 guarantees that $b(f^{k-1})$ is a hardcore function of $f'(x, h_1, \ldots, h_{k-1}) = (f^k(x, h_1, \ldots, h_{k-1}), h_1, \ldots, h_{k-1})$. In the following we show that any algorithm that breaks the pseudorandomness of G too well, violates the hardness of $b(f^{k-1})$ for some $k \in [n+1]$.

Yao [Yao82] showed using a hybrid argument that it is, up to linear factor, as hard to distinguish a pseudorandom sequence from a random one, as it is to predict the next bit of the sequence for every prefix of the sequence. Since it is as hard to distinguish the sequence $(r, \overline{h}, b(f^{n+1}), \ldots, b(f^1))$ from random as it is to distinguish the output of G (for ease of notation we omit the input $(x, \overline{h})$ to the f^k's and the input r from the hardcore bits), by [Yao82] it suffices to show that for every $k \in [n+1]$ it is hard to predict $b(f^{k-1})$ given $(r, \overline{h}, b(f^{n+1}), \ldots, b(f^k))$. Assume toward a contradiction the

existence of an efficient algorithm A for which $\Pr[A(r,\overline{h},b(f^{n+1}),\ldots,b(f^k)) = b(f^{k-1})] > \frac{1}{2} + \text{neg}(n)$. Consider the following efficient algorithm M^A for predicting $b(f^{k-1})$ given $(r,\overline{h},f^k(x,\overline{h}))$.

ALGORITHM 5.3.9

Predictor M^A.
Input: $(r,h_1,\ldots,h_{k-1},f^k(x,h_1,\ldots,h_{k-1}))$.

1. Choose uniformly at random $h_k,\ldots,h_n \in \mathcal{H}$,
2. Generate $f^{k+1},\ldots,f^{n+1}$ from $(f^k,h_{k+1},\ldots,h_n)$.
3. Output $A(r,h_1\ldots,h_n,b(f^{n+1}),\ldots,b(f^k))$

By choosing $h_{k+1},\ldots,h_n$ independently at random, M^A generates a series $(f^1,\ldots,f^{n+1})$ that has the same distribution as in the evaluation of G. Thus, the procedure M^A succeeds in predicting $b(f^{k-1})$ with probability at least $\frac{1}{2}+\text{neg}(n)$, and a contradiction is derived. □

From any hardness

The next theorem generalizes Theorem 5.3.8 for regular one-way function with arbitrary hardness. Since it follows the same lines as the proof of Theorem 5.3.8, the proof of the next theorem is omitted.

THEOREM 5.3.10

Let $f:\{0,1\}^n \mapsto \{0,1\}^n$ be a regular $t(n)$-one-way function and let $\mathcal{H}$ be an efficient family of pairwise-independent length preserving hash functions. Define $G:\{0,1\}^n \times \mathcal{H}^{d-1} \times \{0,1\}^{2n} \mapsto \{0,1\}^{d\ell} \times \mathcal{H}^{d-1} \times \{0,1\}^{2n}$ as

$$G(x,\overline{h},r) = (gl_\ell(f^1(x,\overline{h}),r),\ldots,gl_\ell(f^d(x,\overline{h}),r),\overline{h},r),$$

where:

- $\ell = \left\lfloor \frac{c}{4}\cdot\log(t(n))\right\rfloor$ and $d = \lceil n/\ell\rceil + 1$, where c is the constant that appears in Theorem 5.2.2,[11]
- gl_ℓ is the Goldreich-Levin hardcore function (see Theorem 5.2.2),
- Recall that $f^1(x,\overline{h}) = f(x)$ and for all $i \in [d-1]$ it holds that $f^{i+1}(x,\overline{h}) = f(\overline{h}_i(f^i(x,\overline{h})))$.

There exists a polynomial p such that G is a $t'(n)$-pseudorandom generator for any t' satisfying $p(n,t'(n)) < t(n)$. The input length of G is $\Theta(\frac{n^2}{\log t(n)})$ and it stretches its input by $\Omega(\log t(n))$.

5.3.4 An Almost-Linear-Input Construction from a Regular One-way Function

Assuming that the underlying function is one-way in the usual sense (i.e., of super-polynomial hardness), the pseudorandom generator presented in the previous section (when using Theorem 5.3.10) stretches a seed of length $\Theta(n^2/\log(n))$ by $\log(n)$ bits. Although this is an improvement over the GKL generator, it still translates to a rather high loss of security, since the security of the generator on m bits relies on the security of regular one-way function on $\sqrt{m}$ bits. In this section we give a

[11] If f is a one-way function in the usual sense (i.e., of super-polynomial hardness), we let $\ell = \lfloor\log(n)\rfloor$.

modified construction of the pseudorandom generator of Theorem 5.3.8 that takes a seed of length only $m = \Theta(n \log n)$.[12]

Notice that the input length of the generator of Theorem 5.3.8 is dominated by the description of the n independent hash functions $\overline{h} = (h_1, \ldots, h_n)$. The idea of the new construction is to give a derandomization of the choice of the n hash functions. Thus $h_1, \ldots, h_n$ are no longer chosen independently, but are chosen in a way that is sufficient for the proof to go through. The derandomization uses generators against bounded-space distinguishers, and specifically we can use the generator of Nisan [Nis92], (or that of Impagliazzo, Nisan and Wigderson [INW94]). An important observation is that calculating the randomized iterate of an input can be viewed as a bounded-space algorithm, alternatively presented here as a bounded-width layered branching-program. More accurately, at each step the branching program gets a random input h_i and produces $f^{i+1} = f(h_i(f^i))$. We will show that indeed when replacing $h_1, \ldots, h_n$ with the output of a generator that fools related branching programs, then the proof of security still holds (and specifically the proof of Lemma 5.3.5).

THEOREM 5.3.11

Let $f : \{0,1\}^n \mapsto \{0,1\}^n$ be a regular one-way function and let $\mathcal{H}$ be an efficient family of pairwise-independent length preserving hash functions. Let $v(\mathcal{H})$ be the description length of $h \in H$ and let $\mathrm{BSG} : \{0,1\}^{\tilde{n} \in \Theta(n \log n)} \mapsto \{0,1\}^{nv(\mathcal{H})}$ be a bounded-space generator that 2^{-n}-fools every $(2n, n, v(\mathcal{H}))$-LBP.[13] Define $G' : \{0,1\}^n \times \{0,1\}^{\tilde{n}} \times \{0,1\}^{2n} \mapsto \{0,1\}^{n+1+\tilde{n}+2n}$ as

$$G'(x, s, r) = (b(f^1(x, \mathrm{BSG}(s)), r), \ldots, b(f^{n+1}(x, \mathrm{BSG}(s)), r), \overline{h}, r),$$

where:

- b is the Goldreich-Levin hardcore predicate (see Theorem 5.2.2),
- Recall that $f^1(x, \overline{h}) = f(x)$ and for $2 \leq i \leq n+1$ it holds that $f^i(x, \overline{h}) = f(\overline{h}_{i-1}(f^{i-1}(x, \overline{h})))$.

Then, G is a pseudorandom generator.

Proof outline. The proof of the derandomized version follows in the steps of the proof of Theorem 5.3.10. We give a high-level outline of this proof, focusing only on the main technical lemma that changes slightly. The proof first shows that given the k'th randomized iterate $f^k(x, \overline{h})$ and $\overline{h}$ it is hard to compute $f^{k-1}(x, \overline{h})$ (analogously to Lemma 5.3.2), only now this also holds when the hash functions are chosen as the output of the bounded-space generator. The proof is identical to the proof of 5.3.2, only replacing appearances of $\overline{h}$ with the seed s. Again, the key to the proof is the following technical lemma (slightly modified from Lemma 5.3.5).

LEMMA 5.3.12

For every set $T \subseteq \mathrm{Im}(f) \times \{0,1\}^{\tilde{n}}$, if

$$\Pr[(f^k(U_n, \mathrm{BSG}(U_{\tilde{n}})), U_{\tilde{n}}) \in T] \geq \gamma \ ,$$

[12] Alternatively, we could apply the same modification to the pseudorandom generator of Theorem 5.3.10. The final result, however, would be the same while the resulting construction would be somewhat more complicated.

[13] We mention that the existence of such a generator follows by Theorem 5.2.8.

then

$$\Pr[(f(U_n), U_{\tilde{n}}) \in T] \geq \gamma^2/(k+1) \ .$$

Once we know that $f^{k-1}(x, \mathrm{BSG}(s))$ is hard to compute given $f^k(x, \mathrm{BSG}(s))$ and s, we deduce that one cannot predict a hardcore-bit $b(f^{k-1}(x, \mathrm{BSG}(s)), r)$ given $f^k(x, \mathrm{BSG}(s))$ and the seed s to the bounded-space generator. From here, the proof follow just as the proof of Theorem 5.3.10 in showing that the output of G' is an unpredictable sequence and therefore a pseudorandom sequence.

Proof. (of Lemma 5.3.12) Denote by $g : \{0,1\}^n \times \{0,1\}^{\tilde{n}} \mapsto \mathrm{Im}(f) \times \{0,1\}^{\tilde{n}}$ the function taking inputs of the form (x, s) to outputs of the form $(f^k(x, \mathrm{BSG}(s)), s)$. We proceed by giving bounds on the collision-probability of g. For every two inputs to g, $(x_0, \tilde{h_0})$ and $(x_1, \tilde{h_1})$, in order to have a collision we must first have that $\tilde{h_0} = \tilde{h_1}$ which happens with probability $1/2^{\tilde{n}}$. Now, given that $\tilde{h_0} = \tilde{h_1} = s$ (with a random s), we analyze the probability of the event that $f^k(x_0, \mathrm{BSG}(s))$ equals $f^k(x_1, \mathrm{BSG}(s))$. Consider the following $(2n, n, v(\mathcal{H}))$-LBP for the input pair (x_0, x_1):

- Input: The LBP starts at a node labeled by (x_0, x_1).
- Layer 1: move to the node $(f(x_0), f(x_1))$.
- Layer $i+1$ (for $i \in [k]$): Get random input $h_i \in \mathcal{H}$ and move from node (x_0^i, x_1^i) to the node $(f(h_i(x_1^i)), f(h_i(x_0^i)))$.
- Let (x_0^{k+1}, x_1^{k+1}) be the final node, the LBP accepts if $x_0^{k+1} = x_1^{k+1}$ and rejects otherwise.

The LBP described above has parameters $s = 2n$, $t = n$ and $v = 2n$. Furthermore, it accepts with probability that is exactly the desired collision probability, that is, the probability that $f^k((x_0, \overline{h})) = f^k((x_1, \overline{h}))$ over *any* distribution on $\overline{h} = (h_1, \ldots, h_{k-1})$. For every pair (x_0, x_1) with $f(x_0) \neq f(x_1)$ this probability over random $\overline{h}$ was bounded in the proof of Lemma 5.3.5 by:

$$\Pr_{\overline{h} \leftarrow \mathcal{H}^{k-1}}[f^k(x_0, \overline{h}) = f^k(x_1, \overline{h})] \leq \frac{k-1}{|\mathrm{Im}(f)|}$$

Since the generator fools the above LBP, then replacing the random inputs $\overline{h}$ with the output of the bounded-space generator does not change the probability of acceptance by more than $\varepsilon_A(n) = 2^{-n}$. Therefore, assuming $f(x_0) \neq f(x_1)$, we have that

$$\Pr_{s \leftarrow U_{\tilde{n}}}[f^k(x_0, \mathrm{BSG}(s)) = f^k(x_1, \mathrm{BSG}(s))] \leq \frac{k-1}{|\mathrm{Im}(f)|} + \frac{1}{2^n} \leq \frac{k}{|\mathrm{Im}(f)|}$$

When taking the probability over random (x_0, x_1) we also add the probability that $f(x_0) = f(x_1)$. Thus

$$\Pr_{x_0, x_1 \leftarrow U_n, s \leftarrow U_{\tilde{n}}}[f^k(x_0, \mathrm{BSG}(s)) = f^k(x_1, \mathrm{BSG}(s))] \leq \frac{k}{|\mathrm{Im}(f)|} + \frac{1}{|\mathrm{Im}(f)|} = \frac{k+1}{|\mathrm{Im}(f)|}$$

When the above is plugged into the calculation of the collision-probability of the function g (recall that $g(x, s) = (f^k(x, \mathrm{BSG}(s)), s)$), we get:

$$\mathrm{CP}(g(U_n, U_{\tilde{n}})) \leq \frac{k+1}{2^{\tilde{n}} \cdot |\mathrm{Im}(f)|} \tag{5.3}$$

On the other hand, we check the probability of getting a collision inside the set T. We first request that both $(x_0, \tilde{h_0}) \in T$ and $(x_1, \tilde{h_1}) \in T$, which happens with probability at least γ^2. Then once inside T, we know that the probability of collision is at least $1/|T|$. Altogether:

$$\mathrm{CP}(g(U_n, U_{\tilde{n}})) \geq \gamma^2 \cdot \frac{1}{|T|} \tag{5.4}$$

Combining (5.3) and (5.4) we get

$$\frac{|T|}{2^{\tilde{n}} \cdot |\mathrm{Im}(f)|} \geq \frac{\gamma^2}{k+1} \ .$$

But the probability of getting a value in T when choosing a random element in $\mathrm{Im}(f) \times \{0,1\}^{\tilde{n}}$ is exactly $\frac{|T|}{2^{\tilde{n}} \cdot |\mathrm{Im}(f)|}$. Thus, $\Pr[(y,s) \in T] \geq \gamma^2/(k+1)$ as requested. □

REMARK 5.3.13

It is tempting to think that one should replace Nisan/INW generator in the above proof with the generator of Nisan and Zuckerman [NZ96]. That generator may have seed of size $\Theta(n)$ (rather than $\Theta(n \log n)$) when $s = 2n$ as in our case. Unfortunately, with such a short seed, that generator will incur an error $\varepsilon_A(n) = 2^{-n^{1-\gamma}}$ for some constant γ, which is too high for our proof to work. In order for the proof to go through we need that $\varepsilon_A(n) < poly(n)/|\mathrm{Im}(f)|$. Interestingly, this means that we get a linear-input construction when the image size is significantly smaller than 2^n. In order to achieve a linear-input construction in the general case, we need better generators against LBPs (that have both short seed and small error).

Dealing with Non Length-preserving Functions

The pseudorandom generators presented in this section assumed that the underlying regular one-way function is length-preserving. We mention that this is not a necessity and outline how any regular one-way function can be used. For the simple case that f is shrinking, simply padding the output to the same length is sufficient. The more interesting case is of a length-expanding one-way function f. The important point is that we want the generator to be almost linear in the length of the input to f rather than its output. In Lemma 2.3.4 we show how to transform an expanding one-way function f from $\{0,1\}^n$ to $\{0,1\}^{\ell(n)}$ (for simplicity we write just ℓ) into a length preserving one-way function from $\{0,1\}^{2n}$ to $\{0,1\}^{2n}$. This construction, however, does not maintain the regularity of the one-way function (it maintains only an approximate regularity).

For the regular case we suggest a different solution. Rather than changing the underlying one-way function to be length preserving, we change the randomizing hash functions to be shrinking. That is, given a regular one-way function $f : \{0,1\}^n \mapsto \{0,1\}^\ell$, define the randomized iterate of this function with respect to a family of hash functions from $\{0,1\}^\ell$ to $\{0,1\}^n$. The randomized iterate is now well defined, and moreover, we can show that the collision probability hardly changes. Previously the only way to introduce a new collision was during the application of f (a collision happened with probability $1/\mathrm{Im}(f)$ in each iteration). In the new construction a collision can also be introduced when applying a hash function. But such a collision during hashing happens with probability only 2^{-n} (by the pairwise independence of the hash). Thus the overall collision probability at most doubles and the proof of security follows.

The only problem with this approach is that the description of such hash functions is too long (it is $\Theta(\ell)$ instead of $\Theta(n)$). This is overcome by using an efficient family of *almost* pairwise-

independent hash functions from ℓ bits to n bits with error 2^{-n} (see Definition 2.2.5), which requires a description of only $\Theta(n)$ bits.

5.4 Pseudorandom Generator from an Exponentially Hard One-way Function

5.4.1 Overview

The randomized iterate and general one-way functions

As mentioned in the introduction, the last randomized iteration of a general one-way function is not necessarily hard to invert and in fact may be easy to invert. However, this hardness is not totally diminished, it simply deteriorates in every additional iteration. By refining the techniques used in the case of regular one-way functions we manage to give a lower bound on this deterioration of the hardness to invert the function. More precisely, we show in Section 5.4.2 that there exists a set S^k of inputs to f^k with density at least $\frac{1}{k}$ such that the k'th randomized iteration is hard to invert over inputs taken from S^k. As a result, a hard core bit of the k'th randomized iteration has the guarantee that with probability at least $\frac{1}{k}$ it is pseudorandom.

The multiple randomized iterate

Our goal is to get a string of pseudorandom bits, and the idea is to run m independent copies of the randomized iterate (on m independent inputs). We call this the ***multiple randomized iterate***. From each of the m copies we output a hardcore bit of the k'th iteration. This forms a string of m bits, of which $\frac{m}{k}$ are expected to be random looking. The next step is to run a randomness extractor on such a string (where the output of the extractor is of length, say, $\frac{m}{2k}$). This ensures that with very high probability, the output of the extractor is a pseudorandom string of bits.

The pseudorandom generator - a first attempt

A first attempt for the pseudorandom generator runs the multiple randomized iterate (on m independent inputs) for d (to be determined later) iterations. For each $k \in [d]$ we extract $\frac{m}{2k}$ bits at the k'th iteration. These bits are guaranteed to be pseudorandom (even when given all of the values at the $(k+1)$'th iterate and all of the randomizing hash functions). Thus outputting the concatenation of the pseudorandom strings for the different values of k forms a long pseudorandom output (by a standard hybrid argument).

This concatenation, however, is still not long enough. It is required that the output of the generator is longer than its input, which is not the case here. The input contains m strings $x_1, \ldots, x_m$ and md hash functions. The hash functions are included in the output, so the rest of the output needs to make up for the mn bits of $x_1, \ldots, x_m$. At each iteration we output $\frac{m}{2k}$ bits which adds up to $\sum_{k=1}^{d} \frac{m}{2k}$ bits. This is a harmonic progression that is bounded by $m\frac{\log d}{2}$ and in order to exceed the mn lost bits of the input, we need $d > 2^n$ which is far from being efficient.

The pseudorandom generator and exponential hardness

The failed generator from above can be remedied when the exponential hardness comes into play. It is known that if a function is 2^{cn}-one-way (for some constant $c \in (0,1)$), then it has a $2^{c'n}$-hardcore

function of $c'n$ bits (for another constant c'). Thus, if the original hardness was exponential, then in the k'th iteration we can actually extract $c'n$ random looking strings, each of length $\frac{m}{2k}$. Altogether we get that the output length is $c'n\sum_{k=1}^{d}\frac{m}{2k} \geq c'mn\log d$. Thus for a choice of d such that $\log d > c'$, we get that the overall output is a pseudorandom string of length greater than the input. The input length of the construction is $\Theta(nm)$, where m can be taken to be approximately $\Theta(\log\frac{d}{\varepsilon(n)})$ where $\varepsilon(n)$ is the security of the resulting generator. In particular, in order to get a $2^{\Omega(n)}$-pseudorandom-generator one needs to take a seed of length $\Theta(n^2)$.

To sum up, we describe the full construction in a slightly different manner: One first creates a matrix of size $m \times d$, where each *row* in the matrix is generated by computing the first d randomized iterates of f (each row takes independent inputs). Now from each entry in the matrix $\Theta(cn)$ hardcore bits are computed (thus generating a matrix of hardcore bits). The final stage runs a randomness extractor on each of the *columns* of the hardcore bits matrix.[14] Moreover, the number of pseudorandom bits extracted from a column deteriorates from one iteration to another ($\frac{m}{k}$ pseudorandom bits are taken at the columns associated with the k'th randomized iterate).

Some Remarks

- Our method also works for $2^{\phi(n)}$-one-way functions as long as $\phi \in \Omega(\frac{n}{\log n})$. Loosely speaking, this is because for large values of d, the value $\frac{1}{d}$ becomes too small to overcome with limited repetition (and thus requires m to grow substantially).
- The description in this section focuses on length-preserving one-way functions, the results can be generalized, using Lemma 2.3.4, to use non-length preserving functions.

5.4.2 The Last Randomized Iterate is (sometimes) Hard to Invert

We now formally state and prove the key observation, that there exists a set of inputs of significant weight for which it is hard to invert the k'th randomized iteration even if given access to all of the hash functions leading up to this point.

LEMMA 5.4.1
Let $f\colon \{0,1\}^n \mapsto \{0,1\}^n$ be a $(t(n),\delta(n))$-one-way, let $k \in \text{poly}(n)$, and let f^k and $\mathcal{H}$ be as in Definition 5.3.1. Finally, let

$$S^k \stackrel{\text{def}}{=} \left\{(x,\overline{h}) \in \{0,1\}^n \times \mathcal{H}^{k-1} \mid D_f(f^k(x,\overline{h}) = \max_{j\in[k]} D_f(f^j(x,\overline{h}))\right\} \ .$$

Then there exists a polynomial p, such that for every algorithm A of running-time at most $(t(n)-p(n))$ it holds that

$$\Pr_{(x,\overline{h})\leftarrow S^k}[A(f^k(x,\overline{h}),\overline{h}) = f^{k-1}(x,\overline{h})] \leq \sqrt[3]{16nk^2\delta(n)} \ ,$$

where the probability is also taken over the random coins of A.

[14] Note that each execution of the extractor runs on a column in which each entry consists of a single bit (rather than $\Theta(cn)$ bits). This is a requirement of the proof technique.

Proof. The proof follows similar lines to the proof of Lemma 5.3.2. We start by showing that S^k is not too small. By the pairwise independence of the randomizing hash functions $\overline{h} = (h_1, \ldots, h_{k-1})$ we have that for each $i \in [k]$, the value $f^i(x, \overline{h})$ is independently and randomly chosen from the distribution $f(U_n)$. Thus, simply by a symmetry argument, the k'th (last) iteration has the heaviest preimage size with probability at least $\frac{1}{k}$. Hence,

$$\Pr[(U_n, H^{k-1}) \in S^k] \geq \frac{1}{k} \tag{5.5}$$

Let A be an algorithm with running-time $t_A(n)$ such that

$$\Pr_{(x,\overline{h}) \leftarrow S^k}[A(f^k(x,\overline{h}),\overline{h})) = f^{k-1}(x,\overline{h})] \geq \varepsilon_A(n) \ ,$$

where $\varepsilon_A(n) > \sqrt[3]{16nk^2\delta(n)}$. We show that unless t_A is large, we can use A in order to violate the hardness of f. Consider the procedure M^A for this task.

ALGORITHM 5.4.2

Algorithm M^A for inverting f.
Input: $y \in \mathrm{Im}(f)$.

1. Choose uniformly at random $\overline{h} \in \mathcal{H}^{k-1}$.
2. Apply $A(y, \overline{h})$ to get an output x.
3. Output $\overline{h}_{k-1}(x)$.

Letting $p(n)$ be the sampling and evaluation time of $\overline{h}$, we have that the running-time of M^A is at most $t(n)$. The rest of the proof of Lemma 5.4.1 shows that M^A succeeds with probability at least $\varepsilon_A(n)^3/16nk^2 > \delta(n)$ on uniformly chosen $y \in \mathrm{Im}(f)$, and we conclude that $t_A(n) > t(n) - p(n)$.

We start by focusing our attention only on those inputs for which A succeeds reasonably well. The following Markov argument implies that the probability of getting an element in the set that A succeeds on is not very small.

CLAIM 5.4.3

Let $T_A \subseteq \mathrm{Im}(f^k) \times \mathcal{H}^{k-1}$ be the subset defined as

$$T_A = \left\{(y,\overline{h}) \in \mathrm{Im}(f^k) \times \mathcal{H}^{k-1} \colon \Pr[f(\overline{h}_{k-1}(A(y,\overline{h})) = y] > \varepsilon_A(n)/2\right\} \ ,$$

then

$$\Pr[(f^k(U_n, H^{k-1}), H^{k-1}) \in T_A \wedge (U_n, H^{k-1}) \in S^k] \geq \varepsilon_A(n)/2k \ .$$

Proof. A simple Markov argument (see the proof of Claim 5.3.4) shows that

$$\Pr_{(x,\overline{h}) \leftarrow S^k}[A(f^k(x,\overline{h}),\overline{h})) = f^{k-1}(x,\overline{h}) \wedge f^k(x,\overline{h}),\overline{h}) \in T_A] \geq \varepsilon_A(n)/2 \ ,$$

and since S^k is not too small, the proof of the claim follows. □

Now that we identified a heavy subset of the inputs that A succeeds upon, we want to say that M^A has a fair chance to hit outputs induced by this subset. This is formally shown in the following lemma.

LEMMA 5.4.4
For every set $T \subseteq \mathrm{Im}(f^k) \times \mathcal{H}^{k-1}$, if

$$\Pr[(f^k(U_n, H^{k-1}), H^{k-1}) \in T \wedge (U_n, H^{k-1}) \in S^k] \geq \delta \ ,$$

then

$$\Pr[(f(U_n), H^{k-1}) \in T] \geq \frac{\delta^2}{2kn} \ .$$

Assuming Lemma 5.4.4 we may conclude the proof of Lemma 5.4.1. By Lemma 5.4.4 (taking $T = T_A$ and $\delta = \varepsilon_A(n)/2k$) yields that $\Pr[(f(U_n), H^{k-1}) \in T_A] \geq \varepsilon_A(n)^2/8nk^3$. Thus M^A has a $\varepsilon_A(n)^2/8nk^3$ chance of hitting the set T_A on which it will succeed with probability at least $\varepsilon_A(n)/2$. Altogether, M^A succeeds in inverting f with probability $\varepsilon_A(n)^3/16nk^3 > \delta(n)$. □

Proof. (of Lemma 5.4.4) Divide the outputs of the function f into n slices according to their preimage size. The set T is divided accordingly into n subsets. For every $j \in [n]$ define the j'th slice $T_j = \left\{(z, \overline{h}) \in T \mid D_f(z) = j\right\}$. We divide S^k into corresponding slices as well, define the j'th slice as $S^k_j = \left\{(x, \overline{h}) \in S^k \mid D_f(f^k(x, \overline{h})) = j\right\}$ (note that since $S^k_j \subseteq S^k$ for each $(x, \overline{h}) \in S^k_j$, for each $0 \leq r < k$ it holds that $D_f(f^r(x, \overline{h})) \leq D_f(f^k(x, \overline{h})) = j$). The proof of Lemma 5.4.4 follows the methodology of the analogous lemma for the regular case (Lemma 5.3.5). More precisely, the proof studies the collision-probability of f^k, only here we look at f^k when restricted to S^k_j (that is, we work separately on each slice). Denote this as:

$$\begin{aligned} & \mathrm{CP}(f^k(U_n, \mathcal{H}^{k-1}) \wedge S^k_j) \\ = \ & \Pr[(f^k(x_0, \overline{h}_0), \overline{h}_0) = (f^k(x_1, \overline{h}_1), \overline{h}_1) \wedge (x_0, \overline{h}_0), (x_1, \overline{h}_1) \in S^k_j] \ , \end{aligned}$$

where $(x_0, \overline{h}_0)$ and $(x_1, \overline{h}_1)$ are uniformly, and independently, chosen in $\{0,1\}^n \times \mathcal{H}^{k-1}$. We first give an upper-bound on this collision-probability.

CLAIM 5.4.5

$$\mathrm{CP}(f^k(U_n, \mathcal{H}^{k-1}) \wedge S^k_j) \leq \frac{k}{|\mathcal{H}|^{k-1} \, 2^{n-j}}$$

Proof. For every two inputs $(x_0, \overline{h}_0)$ and $(x_1, \overline{h}_1)$, in order to have a collision we must first have that $\overline{h}_0 = \overline{h}_1$ which happens with probability $(1/|\mathcal{H}|)^k$. Given that $\overline{h}_0 = \overline{h}_1 = \overline{h}$ (with $\overline{h} \in \mathcal{H}^{k-1}$ being uniform), we require also that $f^k(x_0, \overline{h})$ equals $f^k(x_1, \overline{h})$.

If $f(x_0) = f(x_1)$ then a collision is assured. Since it is required that $(x_0, \overline{h}_0) \in S^k_j$ it holds that $D_f(f(x_0)) \leq D_f(f^k(x_0, \overline{h})) = j$ and therefore $\left|f^{-1}(f(x_0))\right| \leq 2^j$. Thus, the probability for that $x_1 \in f^{-1}(f(x_0))$ (and thus of $f(x_0) = f(x_1)$) is at most 2^{j-n}. Otherwise, there must be an $j \in [k-1]$ for which $f^j(x_0, \overline{h}) \neq f^j(x_1, \overline{h})$, but $f^{j+1}(x_0, \overline{h}) =$

$f^{j+1}(x_1, \overline{h})$. Since $f^j(x_0, \overline{h}) \neq f^j(x_1, \overline{h})$, then due to the pairwise-independence of $\mathcal{H}$, the values $\overline{h}_j(f^j(x_0, \overline{h}))$ and $\overline{h}_j(f^j(x_1, \overline{h}))$ are uniformly random values in $\{0,1\}^n$, and thus $f(\overline{h}_j(f^j(x_0, \overline{h}))) = f(\overline{h}_j(f^j(x_1, \overline{h})))$ also happens with probability at most 2^{j-n}. Altogether,

$$\mathrm{CP}(f^k(U_n, \mathcal{H}^{k-1}) \wedge S_j^k) \leq \frac{1}{|\mathcal{H}|^{k-1}}(k+1)2^{j-n} = \frac{k}{|\mathcal{H}|^{k-1} \cdot 2^{n-j}} \ .$$

□

On the other hand, we give a lower-bound for the above collision-probability. We seek the probability of getting a collision inside S_j^k and further restrict our calculation to collisions whose output lies in the set T_j (this further restriction may only reduce the collision probability and thus the lower bound holds also without the restriction). For each slice, denote $\delta_j = \Pr[(f^k(x, \overline{h}), \overline{h}) \in T_j \wedge (x, \overline{h}) \in S_j^k]$. In order to have this kind of collision, we first request that both inputs are in S_j^k and generate outputs in T_j, which happens with probability δ_j^2. Then once inside T_j we require that both outputs collide, which happens with probability at least $\frac{1}{|T_j|}$. Altogether:

$$\mathrm{CP}(f^k(U_n, \mathcal{H}^{k-1}) \wedge S_j^k) \geq \frac{\delta_j^2}{|T_j|} \tag{5.6}$$

Combining Claim 5.4.5 and Equation 5.6 we get:

$$\frac{|T_j| \cdot 2^{j-n-1}}{|\mathcal{H}|^{k-1}} \geq \frac{\delta_j^2}{2k} \tag{5.7}$$

However, note that when taking a random output z and independent hash functions $\overline{h}$, the probability of hitting an element in T_j is at least $2^{j-n-1}/|\mathcal{H}|^{k-1}$ (since each output in T_j has preimage at least 2^{j-1}). This means that $\Pr[(z,h) \in T_j] \geq |T_j| \cdot 2^{j-n-1}/|\mathcal{H}|^{k-1}$ and by Equation (5.7) we deduce that $\Pr[(z,h) \in T_j] \geq \frac{\delta_j^2}{2k}$. Finally, the probability of hitting T is $\Pr[(z,h) \in T] = \sum_j \Pr[(z,h) \in T_j] \geq \sum_j \delta_j^2 2k$. Since $\sum_j \delta_j^2 \geq (\sum_j \delta_j)^2/n$ and (by definition) $\sum_j \delta_j = \delta$, it holds that $\Pr[(z,h) \in T] \geq \frac{\delta^2}{2kn}$ as claimed. □

A Hardcore Function for the Randomized Iterate

A hardcore function of the k'th randomized iteration is taken as a simple generalization of the Goldreich-Levin hardcore function [GL89]. The number of bits taken in this construction depends on the hardness of the function at hand (that is on the hardness of inverting the last iteration of the randomized iterate). Thus, combining Lemma 5.4.1 regarding the hardness of inverting the last iteration and Theorem 5.2.2, we get the following lemma.

LEMMA 5.4.6

Let $f : \{0,1\}^n \mapsto \{0,1\}^n$ be a $t(n)$-one-way function, let $k \in \mathrm{poly}(n)$, and let f^k and S^k be as in Lemma 5.4.1. Let $\{gl_i\}$ and c be as in Theorem 5.2.2 and let $\ell_k = \lfloor \frac{c}{4}(\log(t(n)) - \log(k)) \rfloor$.

Then there exist a polynomial p such that the following holds. The function $hc^k : \mathcal{D}(f^k) \times \{0,1\}^{2n} \mapsto \{0,1\}^\ell$ defined as $hc^k(x, \overline{h}, r) = gl_{\ell_k}(f^{k-1}(x, \overline{h}), r)$, is a $t'(n)$-hardcore function of $\widetilde{f^k}(x, \overline{h}, r) = (f^k(x, \overline{h}), \overline{h}, r)$ over S^k, for any t' satisfying $p(n, t'(n)) < t(n)$.

The following corollary will enable us to apply our extraction lemma (Lemma 5.2.2) to the hardcore function hc^k.

COROLLARY 5.4.7
Let f, ℓ_k, hc^k and $\widetilde{f^k}$ be as in Lemma 5.4.6, then there exist a polynomial p such that the following holds. For any constant $\beta \in (0,1)$ and $i \in [\ell_k]$, the predicate $hc^k_i(z) = hc^k(z)_i$ (i.e., the i'th bit of $hc^k(z)$) is a $(t'(n), 1-\frac{\beta}{k})$ hardcore predicate of the function $\widehat{f^k}(x,\overline{h},r) = (\widetilde{f^k}(x,\overline{h},r), hc^k(x,\overline{h},r)_{1,\ldots,i-1})$, for any t' satisfying $p(n,t'(n)) < t(n)$.

Proof. By the pairwise independence of $\mathcal{H}$, we have that for each $0 \leq j \leq k$ the value $f^j(x,\overline{h})$ is independently and randomly chosen from the distribution $f(U_n)$. Thus, simply by a symmetry argument, the k'th (last) iteration has the heaviest preimage size with probability at least $\frac{1}{k}$. It follows that $\Pr[(U_n, H^{k-1}) \in S^k] \geq \frac{1}{k}$, and the proof follows by Lemma 5.4.6. □

5.4.3 The Multiple Randomized Iterate

In this section we consider the function $f^{m\times k}$ which consists of m independent copies of the randomized iterate f^k.

CONSTRUCTION 5.4.8 (the k'th multiple randomized iterate)
Let $m,k \in \mathbb{N}$, and let f^k and $\mathcal{H}$ be as in Definition 5.3.1. We define the k'th Multiple Randomized Iterate $f^{m\times k} : \{0,1\}^{mn} \times \mathcal{H}^{mk} \mapsto \mathrm{Im}(f)^m$ as:

$$f^{m\times k}(\overline{x}, V) = f^k(\overline{x}_1, V_1), \ldots, f^k(\overline{x}_m, V_m),$$

where $\overline{x} \in \{0,1\}^{mn}$ and $V \in \mathcal{H}^{m\times(k-1)}$. We let $H^{m\times j}$ be the random variable uniformly distributed over $\mathcal{H}^{m\times j}$.

For each of the m outputs of $f^{m\times k}$ we look at its hardcore function hc^k. By Lemma 5.4.6 it holds that m/k of these m hardcore strings are expected to fall inside the "hard-set" of f^k (and thus are indeed pseudorandom given $(f^{m\times k}(\overline{x},V),V)$). The next step is to invoke a randomness extractor on a concatenation of one bit from each of the different independent hardcore strings. The output of the extractor is taken to be of length $\lfloor \frac{m}{4k} \rfloor$. The intuition being that with high probability, the concatenation of single bits from the different outputs of hc^k contains at least $\frac{m}{2k}$ pseudoentropy. Thus, the output of the extractor forms a pseudorandom string and might serve as a hardcore function of the multiple randomized iterate $f^{m\times k}$.

LEMMA 5.4.9 (hardcore function for the multiple randomized iterate)
Let $f : \{0,1\}^n \mapsto \{0,1\}^n$ be a $t(n)$-one-way function, let $k \in \mathrm{poly}(n)$, let $f^{m\times k}$ and $hc^k : \mathcal{D}(f^k) \times \{0,1\}^{2n} \mapsto \{0,1\}^{\ell_k}$ be as in Construction 5.4.8 and Lemma 5.4.6 respectively, and let $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^m \mapsto \{0,1\}^{\lfloor \frac{m}{4k} \rfloor}$ be a $(\lfloor \frac{m}{2k} \rfloor, \varepsilon_k)$-strong extractor. We define $hc^{m\times k} : \mathcal{D}(f^{m\times k}) \times \{0,1\}^{3n} \mapsto \{0,1\}^{\ell_k \cdot \lfloor \frac{m}{4k} \rfloor}$ as $hc^{m\times k}(\overline{x},V,r,y) = (w_1(\overline{x},V,r,y), \ldots, w_d(\overline{x},V,r,y))$, where $\overline{x} \in \{0,1\}^{mn}$, $V \in \mathcal{H}^{m\times(k-1)}$, $r \in \{0,1\}^{2n}$ and $y \in \{0,1\}^n$, and for every $i \in [\ell_k]$

$\widetilde{w_i}(\overline{x}, V, r, y) \stackrel{\text{def}}{=} \text{Ext}(y, (hc^k(\overline{x}_1, V_1), r)_i, \ldots, (hc^k(\overline{x}_m, V_m), r)_i))$. Finally, we let $\widetilde{f^{m\times k}}(\overline{x}, V, r, y) = (f^{m\times k}(\overline{x}, V), V, r, y)$

Then, there exists a polynomial p such that the following holds. For every $\gamma(n) > m \cdot 2^{-n/4}$ and $t'(n)$ satisfying $p(t'(n), n, m, k, 1/\gamma(n)) < t(n)$, the function $hc^{m\times k}$ is a $\ell_k(\varepsilon_k + \rho_k + \gamma))$-hardcore function of $\widetilde{f^{m\times k}}$, where ρ_k is the probability is the probability that when taking m independent samples in $\mathcal{D}(f^k)$, less than $\lfloor \frac{m}{2k} \rfloor$ samples are in a fixed subset of density $2/3k$.

Proof. Let A be an algorithm that given $\widetilde{f^{m\times k}}$, distinguishes between a uniform string and the hardcore function with advantage δ_A. By a standard hybrid argument, there exists an algorithm A' and that runs essentially in the same as A, and for some $j \in [\ell_k]$ distinguishes the value of w_j from random with advantage δ_A/ℓ_k, given $\widetilde{f^{m\times k}}$ and $\left\{hc^k(\overline{x}_s, V_s), r)_i \colon s \in [m], i \in [j-1]\right\}$. The key observation is that given the hardcore properties of hc^k guaranteed by Corollary 5.4.7, Lemma 5.2.2 yields that it is hard to distinguish w_j from random with advantage more than $\varepsilon + \rho_k + \gamma$, for any algorithm of running-time that is not too large. Namely, $p'(t_A(n), n, m, k, 1/\gamma(n)) < t^c(n)$, for some polynomial p' and a constant $c \in (0, 1)$. We conclude that for the right choice of p it holds that $p(t_A(n), n, m, k, 1/\gamma(n)) < t(n)$. □

5.4.4 A Pseudorandom Generator from Exponentially Hard One-way Functions

We are now ready to present our pseudorandom generator. After deriving a hardcore function for the multiple randomized iterate, the generator is similar to the construction from regular one-way function. That is, run randomized iterations and output hardcore bits. The major difference in our construction is that, for starters, it uses hardcore functions rather than hardcore bits. More importantly, the amount of hardcore bits extracted at each iteration is not constant and deteriorates with every additional iteration. The following theorem follows immediately by Lemma 5.4.9 and a standard BMY like indistinguishability argument (see for example the proof of Theorem 5.3.10).

THEOREM 5.4.10 (the pseudorandom generator)
Let $f \colon \{0,1\}^n \mapsto \{0,1\}^n$ be a $t(n)$-one-way function and let $d, m \in \text{poly}(n)$. For $k \in [d]$ let $\text{Ext}_k \colon \{0,1\}^n \times \{0,1\}^m \mapsto \{0,1\}^{\lfloor \frac{m}{4k} \rfloor}$ be a $(\lfloor \frac{m}{2k} \rfloor, \varepsilon_k)$-strong extractor and let $hc^{m\times k} \colon \mathcal{D}(f^{m\times k}) \times \{0,1\}^{3n} \mapsto \{0,1\}^{\ell_k \cdot \lfloor \frac{m}{4k} \rfloor}$ be the hardcore function defined in Lemma 5.4.9 w.r.t. f and Ext_k. We define G as

$$G(\overline{x}, V, r, y) = (hc^{m1}(\overline{x}, V, r, y) \ldots, hc^{md}(\overline{x}, V, r, y), V, r, y) \ ,$$

where $\overline{x} \in \{0,1\}^{mn}$, $V \in \mathcal{H}^{m\times(d-1)}$, $r \in \{0,1\}^{2n}$ and $y \in \{0,1\}^n$.

Then there exists a polynomial p such that the following holds. Assuming that $d \in \text{poly}(n)$ and that G stretches its input, then for every $\gamma(n) > m \cdot 2^{-n/3}$ and $t'(n)$ such that $p(t'(n), n, m, d, 1/\gamma(n)) < t(n)$, G is a $(t'(n), dn \cdot \max_{k\in[d]} \{\varepsilon_k + \rho_k + \gamma\})$-pseudorandom generator, where ρ_k is as in Lemma 5.4.9.

With the appropriate choice of parameters we get the following pseudorandom generators.

COROLLARY 5.4.11
Let $\phi(n) \in \Omega(n/\log(n))$ and let $f \colon \{0,1\}^n \mapsto \{0,1\}^n$ be a $2^{\phi(n)}$-one-way function. Then there exists

a value $d = 2^{\Omega(n/\phi(n))}$ such that the following holds. For any $m \in \text{poly}(n)$ such that $m > 8d$, there exists a choice of $\{\text{Ext}_k\}_{k\in[d]}$ for which the function G of Theorem 5.4.10, w.r.t. f, d, m and $\{\text{Ext}_k\}$, is a $2^{\Omega(\min\{n,m/d^2\})}$-pseudorandom generator. The input length of G is $\Theta(mnd)$ and it stretches its input by $\Omega(m\phi(n)/d)$. In particular, for $\phi(n) \in \Omega(n)$ and $m = n$, we get a generator with quadratic seed length and exponential hardness.

Proof. Note that the input length of G is $m|x| + |V| + |r| + |y|$ and since it is dominated by the description of V it is in $\Theta(mdn)$. On the other hand, the output length of G is $m(\sum_{k=1}^{d} \ell_k \lfloor \frac{m}{4k} \rfloor) + |V| + |r| + |y|$. Thus, G stretches its input by $(\sum_{k=1}^{d} \ell_k \lfloor \frac{m}{4k} \rfloor - mn \geq (\lfloor c \cdot \log(t(n)) \rfloor (\sum_{k=1}^{d}(1 - \log(k)) \lfloor \frac{m}{4k} \rfloor) - mn$, for some universal constant $c \in (0,1)$. Since we consider super-polynomial values for $t(n)$, $d \in \text{poly}(n)$ and $m \geq 8d$, G stretches its input by at least $m(\lfloor \frac{c}{8} \log(t(n)) \rfloor \cdot (\sum_{k=1}^{d} \frac{1}{k}) - n)$. It follows that there exist a universal constant $\gamma > 1$, which depends on the above c, such that G stretches its input by $\Omega(m\phi(n)/d)$, for $d = 2^{\gamma n/\phi(n)}$ and $m > 9d$.

For the security of G, we note that for every $k \in [d]$ it holds that $\rho_k \leq \rho_d$, and by the Chernoff bound it follows that $\rho_k \leq 2^{-\Omega(m/d^2)}$. In addition, for the proper choice of $\{\text{Ext}_k\}$[15] we have that $\text{Ext}_k \leq \text{Ext}_d \leq 2^{-\Omega(\min\{n,m/d\})}$. Thus, Theorem 5.4.10 yields that G is a $2^{\Omega(\min\{n,m/d^2\})}$-pseudorandom generator. □

5.5 Pseudorandom Generator from Any One-way Function

Our implementation of a pseudorandom generator from any one-way function follows the route of [HILL99], but takes a totally different approach in the implementation of its initial step. More precisely, we follow the outline of Holenstein [Hol06a], which gave a new proof to [HILL99] and includes a description and proof of a pseudorandom generator with seed length $\Theta(n^8)$.[16]

The basic building block of the HILL generator is a ***pseudoentropy pair***.[17] A distribution is said to have *pseudoentropy* at least k if it is computationally-indistinguishable from some distribution that has entropy k. Informally, the a pseudoentropy pair is a pair of a function and predicate on the same input with the following property: The pseudoentropy of the predicate's output when given the output of the function is noticeably larger than the real (conditional) entropy of this bit. In their construction, [HILL99] exploit this gap between real entropy and pseudoentropy to construct a pseudorandom generator. We show that the second randomized iterate of a one-way function together with a standard hardcore predicate forms a pseudoentropy pair with better properties than the [HILL99] one. Hence, plugging our pseudoentropy pair as the first step of the HILL construction, results in a better overall construction. Let us now turn to a more formal discussion. We define the pseudoentropy pair as follows.

DEFINITION 5.5.1 (pseudoentropy pair (PEP))
Let $g : \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}$ and $b : \{0,1\}^n \mapsto \{0,1\}$ be polynomial-time computable functions. The pair (g, b) is a $(t(n), \delta(n), \gamma(n))$-PEP if

[15] For example, the leftover hash lemma ([IL89]) yields that $\text{Ext}_k(x,h) = (h(x), h)$, where $\mathcal{H}$ is a family of pairwise independent hash functions from $\{0,1\}^n$ to $\{0,1\}^{\lfloor \frac{m}{4k} \rfloor}$, is a good choice.

[16] In [HILL99], an explicit construction was given only for a pseudorandom generator with seed length $\Theta(n^{10})$. The existence of the more efficient construction was claimed without being presented or proved.

[17] The notion of pseudoentropy pair is implicitly used in [HILL99], and was formally defined in [HHR06].

1. $\mathrm{H}(b(U_n) \mid g(U_n)) \leq \delta(n)$,
2. b is a $(t(n), 1-\delta(n)-\gamma(n))$-hardcore predicate of g.

We write that (g,b) is a $(\delta(n),\gamma(n))$-PEP, it is $(t(n),\delta(n),\gamma(n))$-PEP for every polynomial $t(n)$.

[HILL99] show how to use any one-way function in order to construct a (δ,γ)-PEP, where $\delta \in [0,1]$ is an *unknown* value and γ is a fraction noticeably smaller than $\frac{1}{2n}$ (i.e., smaller than $\frac{1}{2n} - \frac{1}{p(n)}$ for some polynomial p). They then present a construction of a pseudorandom generator using a $(\delta, \frac{1}{\Theta(n)})$-PEP where δ is *known*. To overcome this gap, the HILL generator enumerates all values for δ (up to an accuracy of $\Omega(\frac{1}{n})$), invokes the generator with every one of these values and eventually combines all generators using an XOR of their outputs. This enumeration costs an additional factor of n to the seed length as well as n^3 times more calls to the underlying one-way function.

In Section 5.5.1 we prove that the second randomized iterate of a one-way function can be used to construct a $(\frac{1}{2},\gamma)$-PEP, where γ is a fraction noticeably smaller than $\frac{1}{2n}$.[18] In Section 5.5.2 we show that by combining our PEP with the second part of [Hol06a] construction, we get a pseudorandom generator that is more efficient and has better security[19] than the original construction of [HILL99]/[Hol06a] (the efficiency improves by a factor of n^3 and the security by a factor of n). For comparison, we present the PEP used by [HILL99] in Appendix 5.6.

5.5.1 A Pseudoentropy Pair Based on the Randomized Iterate

Recall that for a given function f, we have defined (Definition 5.3.1) its second randomized iterate as $f^2(x,h) = f(h(f(x)))$. We would like to prove that the second iterate of a one-way function gives rise to a PEP . Indeed, since the randomized iterate maintains some of the hardness of a function (Lemma 5.4.1), we have that with probability $\frac{1}{2} + \Omega(\frac{1}{n})$ it is hard to compute the value of $f(x)$ given the output $f^2(x,h)$. We want to complement this fact by saying that with probability $\frac{1}{2}$ the value of $f(x)$ can be essentially determined from the output. The latter statement is almost true, except that there typically remains a small amount of uncertainty regarding $f(x)$. To overcome this, we add to the output a small amount of random information about $f(x)$. Specifically, we define the *extended randomized iterate* to include some additional random information about $f(x)$. This information (of $\Theta(\log(n)$ bits) is small enough to diminish any entropy left in $f(x)$ (in $\frac{1}{2}$ of the inputs), yet is not significant enough to drastically change its pseudoentropy.

The extended randomized iterate of any one-way function

For simplicity we assume that the one-way function is length-preserving, the adaptation to any one-way function is done via Lemma 2.3.4. We use the following extended version of the second randomized iterate.

[18] Actually, we prove the more general result that $\gamma \in \Omega(\frac{\log t(n)}{n})$, assuming that f is $t(n)$-one-way.

[19] By "security" we mean that the proof of security of the reduction to the underlying one-way function has better parameters, see Section 2.3.2 for a more detailed discussion.

DEFINITION 5.5.2 (the extended randomized iterate)
Let $f : \{0,1\}^n \mapsto \{0,1\}^n$, let $m = \lceil 3\log(n) + 8 \rceil$, and let $\mathcal{H}$ and $\mathcal{H}_E$ be two families of pairwise-independent hash functions from $\{0,1\}^n$ to $\{0,1\}^n$ and $\{0,1\}^n$ to $\{0,1\}^m$ respectively. We define g, the extended randomized iterate of f, as:

$$g(x,h,h_E) \stackrel{\text{def}}{=} (f^2(x,h), h, h_E(f(x)), h_E)$$

where $x \in \{0,1\}^n$, $h \in \mathcal{H}$ and $h_E \in \mathcal{H}_E$. In the following we denote by H and H_E the random variables uniformly distributed over $\mathcal{H}$ and $\mathcal{H}_E$ respectively.

REMARK 5.5.3
We stress that while the role of $\mathcal{H}_E$ in the above construction and the role of $\mathcal{H}$ in the HILL's PEP are syntactically similar (see Appendix 5.6), their actual role is different. In the above construction the purpose of using $\mathcal{H}_E$ is to reveal a small amount of information (i.e., $\Theta(log(n))$) about $f(x)$, which in turn *slightly* reduces the uncertainty of $f(x)$ when given $g(x,h,h_E)$. Thus $\mathcal{H}_E$ is used to widen the gap between the real and the pseudoentropy of $f(x)$ given $g(x,h,h_E)$. This extra information is not of major importance and indeed even without using $\mathcal{H}_E$ (i.e. using $g'(x,h,h_E) \stackrel{\text{def}}{=} (f(h(f(x))),h))$, we can still construct a PEP, though parameters will not be as good. On the other hand, in the HILL's PEP there are settings where the hash function reveals a significant amount of information (i.e., $\Omega(n)$) about the input of the function. Thus, removing the hash function altogether guarantees no gap between the real and the pseudoentropy, and in particular the resulting pair is not likely to be a PEP.

The heart of this section are the following two lemmata. In Lemma 5.5.4 we show that with probability $\frac{1}{2}+\Omega(\frac{\log t(n)}{n})$ it is hard to compute the value of $f(x)$ given a random output $g(x,h,h_E)$. While in Lemma 5.5.6 we show that the value of $f(x)$ is determined w.h.p. by the value of $g(x,h,h_E)$.

LEMMA 5.5.4
Let f be a $t(n)$-one-way function and let g be as in Definition 5.5.2. Then there exists a set $V \subseteq \mathcal{D}(g)$ of density $\frac{1}{2} + \frac{\log t(n)}{3n}$ and a polynomial p such that the following holds. For every algorithm A of running-time $t'(n)$ satisfying $p(n,t'(n)) < t(n)$, it holds that

$$\Pr_{(x,h,h_E)\leftarrow V\times\mathcal{H}_E}[A(g(x,h,h_E)) = f(x)] < \frac{1}{t'(n)} \ ,$$

where the probability is also taken over the random coins of A.

Proof. Let $V \stackrel{\text{def}}{=} \{(x,h,h_E) \in \mathcal{D}(g) : \mathrm{Deg}_f(f(x)) \leq \mathrm{Deg}_f(f^2(x,h) + \frac{\log t(n)}{2n}\}$ and let R^0 and R^1 be distributed according to $\left\lceil \frac{2\cdot \mathrm{Deg}_f(f(U_n))}{\log t(n)} \right\rceil$ and $\left\lceil \frac{2\cdot \mathrm{Deg}_f(f^2(U_n,H))}{\log t(n)} \right\rceil$ respectively. Since $f(U_n)$ and $f^2(U_n,H) = f(H(f(U_n)))$ are two independent instances of a random variable distributed over $\mathrm{Im}(f)$, it follows that R^0 and R^1 are two independent instances of a random variable distributed over $[\lceil 2n/\log t(n) \rceil]$. By symmetry, $\Pr[R^0 \lneq R^1] = \Pr[R^0 \gneq R^1]$ and since the collision-probability of a random variable is minimal when the distribution is uniform, we have that $\Pr[R^0 = R^1] \geq \frac{1}{\lceil 2n/\log t(n) \rceil}$. Combining the above equations yields that $\Pr[R^0 \leq R^1] \geq \frac{1}{2} + \frac{1}{\lceil 2n/\log t(n) \rceil} \geq \frac{1}{2} + \frac{\log t(n)}{3n}$

and thus

$$\frac{|V|}{|\{0,1\}^n \times \mathcal{H} \times \mathcal{H}_E|} = \Pr[\mathrm{Deg}_f(f(U_n)) \leq \mathrm{Deg}_f(f^2(U_n,H)) + \frac{\log t(n)}{2}]$$
$$\geq \Pr[R^0 \leq R^1] \geq \frac{1}{2} + \frac{\log t(n)}{3n} \ .$$

In order to prove the hardness of g over V, let A be an algorithm that runs in time $t_A(n)$ and $\Pr_{(x,h,h_E)\leftarrow V\times\mathcal{H}_E}[A(g(x,h,h_E)) = f(x)] \geq 1/t_A(n)$. Therefore, there exists an efficient algorithm A', with oracle access to A, that computes $f(x)$ with probability at least $\frac{1}{2^m t_A}$, given only $(f^2(x,h),h)$, i.e. A' on input $(f^2(x,h),h)$ chooses a random value z for $(h_E, h_E(f(x)))$ and returns $A(f^2(x,h),h,z)$. The proof Lemma 5.5.4 follows by the next lemma, which is a straight forward generalization Lemma 5.4.1.

LEMMA 5.5.5 (generalization of Lemma 5.4.1)
Let $f : \{0,1\}^n \mapsto \{0,1\}^n$ be a $(t(n),\delta(n))$-one-way, let $k \in \mathrm{poly}(n)$, and let f^k and $\mathcal{H}$ be as in Definition 5.3.1. For any value of $\mathrm{gap} \in [n]$ we let

$$S^{k,\mathrm{gap}} \stackrel{\mathrm{def}}{=} \left\{ (x,\overline{h}) \in (\{0,1\}^n \times \mathcal{H}^k) \mid \mathrm{Deg}_f(f^k(x,\overline{h}) + \mathrm{gap} \geq \max_{j\in[k]} \mathrm{Deg}_f(f^j(x,\overline{h})) \right\} \ .$$

Then there exists a polynomial p, such that for every algorithm A of running-time at most $(t(n) - p(n))$ it holds that

$$\Pr_{(x,\overline{h})\leftarrow S^{k,\mathrm{gap}}}[A(f^k(x,\overline{h}),\overline{h}) = f^{k-1}(x,\overline{h})] \leq 2^{\mathrm{gap}} \cdot \sqrt[3]{32nk^2\delta(n)} \ ,$$

where the probability is also taken over the random coins of A.

Hence, the above lemma (taking $k = 2$ and $\mathrm{gap} = \left\lceil \frac{\log t(n)}{2} \right\rceil$) yields that $p(n,t'(n)) > t(n)$, for the right choice of p. □

LEMMA 5.5.6
Let f be a $t(n)$-one-way function and let g be as in Definition 5.5.2. Then there exists a set $L \subseteq \mathcal{D}(g)$ of density at least $\frac{1}{2} + \frac{1}{4n}$ such that the following holds for every $(x,h,h_E) \in L$.

$$\Pr_{(x',h',h'_E)\leftarrow\mathcal{D}(g)}[f(x') = f(x) \mid g(x',h',h'_E) = g(x,h,h_E)] > 1 - \frac{1}{16n^2} \ .$$

Proof. Let $\mathrm{Heavy} = \{(x,h,h_E) \in \mathcal{D}(g) : \mathrm{Deg}_f(f(x)) \geq \mathrm{Deg}_f(f^2(x,h)\}$. By symmetry (as in the proof of Lemma 5.5.4) it holds that $|\mathrm{Heavy}| / |\mathcal{D}(g)| \geq \frac{1}{2} + \frac{1}{2n}$. The following claim states that we can set L to be most of the elements inside Heavy.

CLAIM 5.5.7

$$\Pr\left[\frac{\Pr[g(U'_n,H',H'_E) = g(U_n,H,H_E) \bigwedge f(U'_n) = f(U_n)]}{\Pr[g(U'_n,H',H'_E) = g(U_n,H,H_E)]} \geq 1 - \frac{1}{16n^2} \mid (U_n,H,H_E) \in \mathrm{Heavy}\right]$$
$$\geq 1 - \frac{1}{4n} \ .$$

Thus, the proof of Lemma 5.5.6 follows by letting $L = \left\{(x,h,h_E) \in \mathcal{D}(g) : \frac{\Pr[g(U_n,H,H_E)=g(x,h,h_E) \bigwedge f(U_n)=f(x)]}{\Pr[g(U_n,H,H_E)=g(x,h,h_E)]} \geq 1 - \frac{1}{16n^2}\right\}$. Note that indeed,

$$\begin{aligned} & \Pr[(U_n,H,H_E) \in L] \\ \geq \; & \Pr\left[(U_n,H,H_E) \in \text{Heavy} \wedge \frac{\Pr[g(U'_n,H',H'_E) = g(U_n,H,H_E) \bigwedge f(U'_n) = f(U_n)]}{\Pr[g(U'_n,H',H'_E) = g(U_n,H,H_E)]} \geq 1 - \frac{1}{16n^2}\right] \\ \geq \; & \frac{1}{2} + \frac{1}{2n} - \frac{1}{4n} = \frac{1}{2} + \frac{1}{4n} \; . \end{aligned}$$

□

Proof. (of Claim 5.5.7) Let (x,h) be a random element inside $\{0,1\}^n \times \mathcal{H}$. The pairwise independence of $\mathcal{H}$ implies that w.h.p. $\frac{\left|\left\{x' \in \{0,1\}^n : f^2(x',h)=f^2(x,h) \wedge f(x') \neq f(x)\right\}\right|}{|\{0,1\}^n|} \leq 2^{\text{Deg}_f(f^2(x,h))-n}$. When considering also a random h_E, it follows that w.h.p $\frac{|\{x' \in \{0,1\}^n : g(x',h,h_E)=g(x,h,h_E) \wedge f(x') \neq f(x)\}|}{|\{0,1\}^n|} \leq 2^{\text{Deg}_f(f^2(x,h))-n-m}$. On the other hand for every $(x,h,h_E) \in$ Heavy it holds that $\frac{\left|f^{-1}(f(x))\right|}{|\{0,1\}^n|} \geq 2^{\text{Deg}_f(f^2(x,h))-n}$, and we conclude that for a random $(x,h,h_E) \in$ Heavy, w.h.p. $\frac{\Pr[g(U_n,H,H_E)=g(x,h,h_E) \bigwedge f(U_n)=f(x)]}{\Pr[g(U_n,H,H_E)=g(x,h,h_E)]} \geq 1 - \frac{1}{16n^2}$.

Let us turn to a more formal discussion. Let $x \in \{0,1\}^n$, $(z_1,z_2) \in \text{Im}(f) \times \{0,1\}^m$ and define $S_{x,z_1,z_2} = \{(h,h_E) \in \mathcal{H} \times \mathcal{H}_E : g(x,h,h_E) = (z_1,h,z_2,h_E)\}$. For $x' \notin f^{-1}(f(x))$, the pairwise independence of $\mathcal{H}$ and $\mathcal{H}_E$ implies that $\Pr_{(h,h_E) \leftarrow S_{x,z_1,z_2}}[g(x,h,h_E) = g(x',h,h_E)] \leq 2^{\text{Deg}_f(z_1)-n-m}$. Therefore,

$$\Pr_{(h,h_E) \leftarrow S_{x,z_1,z_2}}[g(U_n,h,h_E) = g(x,h,h_E) \bigwedge f(U_n) \neq f(x)] < 2^{\text{Deg}_f(z_1)-n-m} \tag{5.8}$$

Let I_{x,h,h_E} be the indicator random-variable that equals 1 if $\Pr[g(U_n,h,h_E) = g(x,h,h_E) \bigwedge f(U_n) \neq f(x)] < 8n \cdot 2^{\text{Deg}_f(f^2(x,h))-n-m}$. By (5.8) and Markov's inequality we get that, $\Pr_{(h,h_E) \leftarrow S_{x,z_1,z_2}}[I_{x,h,h_E} = 1] \geq 1 - \frac{1}{8n}$. The uniform distribution over $\mathcal{H} \times \mathcal{H}_E$ can be viewed as the distribution induced by choosing a random subset $S_{x,z_1,z_2} \subset \mathcal{H} \times \mathcal{H}_E$ according to its density (i.e., S_{x,z_1,z_2} is chosen with probability $|S_{x,z_1,z_2}| / |\mathcal{H} \times \mathcal{H}_E|$) and then choosing a uniform pair (h,h_E) in S_{x,z_1,z_2}. It follows that $\Pr[I_{x,H,H_E} = 1] \geq 1 - \frac{1}{8n}$ (probability here is taken over uniform choice of H and H_E). Recall that $|\text{Heavy}| / |\mathcal{D}(g)| \geq \frac{1}{2} + \frac{1}{2n}$. Thus, by averaging over all $x \in \{0,1\}^n$ we have that $\Pr[I_{U_n,H,H_E} = 1 \mid (U_n,H,H_E) \in \text{Heavy}] \geq 1 - \frac{1}{4n}$ (where probability is taken over the uniform choice of $x \in U_n$ and H, H_E). Since $m = \lceil 3\log(n) + 8 \rceil$, it follows that

$$\begin{aligned} & \Pr\left[\Pr[g(U'_n,H,H_E) = g(U_n,H,H_E) \bigwedge f(U'_n) \neq f(U_n)] < \frac{2^{\text{Deg}_f(f^2(U_n,H))-n}}{32n^2}\right. \\ & \left. \mid \; (U_n,H,H_E) \in \text{Heavy}\right] \geq 1 - \frac{1}{4n} \; . \end{aligned} \tag{5.9}$$

On the other hand, for any $(x,h,h_E) \in \mathcal{D}(g)$ it holds that $\Pr[g(U_n,h,h_E) = g(U'_n,h',h'_E)] \geq 2^{\text{Deg}_f(f^2(U_n,H))-n-1}$ and we conclude that

$$\begin{aligned} & \Pr\left[\frac{\Pr[g(U'_n,H',H'_E) = g(U_n,H,H_E) \bigwedge f(U'_n) \neq f(U_n)]}{\Pr[g(U'_n,H',H'_E) = g(U_n,H,H_E)} \leq \frac{1}{16n^2} \mid (U_n,H,H_E) \in \text{Heavy}\right] \\ \geq \; & 1 - \frac{1}{4n} \; . \end{aligned}$$

□

Constructing the pseudoentropy pair

Recall that the PEP consists of a function and a corresponding predicate. For the function we use the extended randomized iterate, and for the predicate we basically take a Goldreich-Levin hardcore bit. The twist here is that unlike a standard GL predicate, we take the hardcore bit from the intermediate value $f(x)$ rather than from the actual input x. While a hardcore bit taken from x has the required computational hardness, it may also have entropy to it. On the other hand, taking the predicate from $f(x)$ is what gives the desired gap between entropy and pseudoentropy. In the following formal theorem we use a slightly modified version of g to incorporate the randomness required by the hardcore predicate. The function and predicate are proved to form a $(\frac{1}{2}, \frac{\log t(n)}{4n})$-PEP.

THEOREM 5.5.8

Let f be a $t(n)$-one-way function, and let $\mathcal{H}$, $\mathcal{H}_E$ and g be as in Definition 5.5.2. For $r \in \{0,1\}^n$, let $g'(x,h,h_E,r) = (g(x,h,h_E),r)$ and $b(x,h,h_E,r) = \langle f(x), r\rangle_2$. Then there exists a polynomial p such that the pair (g',b) is a $(t'(n), \frac{1}{2}, \frac{\log t(n)}{4n}))$-PEP , for every t' satisfying $p(n,t'(n)) < t(n)$.

Proof. The computational side of the theorem follows directly from Lemma 5.5.4[2] and Theorem 5.2.2 (i.e., there exists a polynomial p for which b is a $(t'(n), \frac{1}{2} + \frac{\log t(n)}{4n}))$-hardcore predicate of g' for any t' satisfying $p(n,t'(n)) < t(n)$). It is left to prove that $\mathrm{H}(b'(W_n,U_n) \mid g'(W_n,U_n)) \leq \frac{1}{2}$, where W_n is uniformly distributed over $\{0,1\}^n \times \mathcal{H} \times \mathcal{H}_E$. By Lemma 5.5.6 there exists a set $L \subseteq \mathcal{D}(g)$ of density $\frac{1}{2} + \frac{1}{4n}$ such that for every $(x,h,h_E) \in L$ it holds that $\Pr_{(x',h',h'_E) \leftarrow \mathcal{D}(g)}[f(x') = f(x) \mid g(x',h',h'_E) = g(x,h,h_E)] > 1 - \frac{1}{16n^2}$. Hence,

$$\begin{aligned}
& \mathrm{H}(b(W_n,U_n) \mid g'(W_n,U_n)) = \mathrm{H}(b(W_n,U_n) \mid g(W_n),U_n)) \\
\leq\ & (\frac{1}{2} + \frac{1}{4n}) \cdot \mathrm{H}(\frac{1}{16n^2}, 1 - \frac{1}{16n^2}) + (\frac{1}{2} - \frac{1}{4n}) \\
<\ & (\frac{1}{2} + \frac{1}{4n}) \left(\frac{1}{16n^2}(1 - \frac{1}{16n^2}) \right)^{1/2} + (\frac{1}{2} - \frac{1}{4n}) < \frac{1}{8n} + \frac{1}{16n^2} + (\frac{1}{2} - \frac{1}{4n}) < \frac{1}{2} \ ,
\end{aligned}$$

where the second inequality is due to the fact that $\mathrm{H}(q,1-q) \leq 2(q(1-q))^{1/\ln(4)}$ (c.f., [Top01, Theorem 1.2]) and since for small enough q (i.e., $q < 1/100$) it holds that $2(q(1-q))^{1/\ln(4)} < (q(1-q))^{1/2}$. □

5.5.2 The Pseudorandom Generator

The following is adapted from [Hol06a, Lemma 5].

PROPOSITION 5.5.9

Let $\gamma(n) > \frac{1}{n}$ be a polynomial computable function and let (g,b) be a $(t(n),\delta(n),\gamma(n))$-PEP. Suppose we are given two non-uniform advices α_n, β_n (for any n) satisfying $\alpha_n \leq \delta(n) \leq \alpha_n + \frac{\gamma(n)}{4}$ and $\beta_n \leq \mathrm{H}(g(U_n)) \leq \beta_n + \frac{\gamma(n)}{4}$. Then there exists a polynomial p such that the following holds for every polynomial computable function $\varepsilon : \mathbb{N} \mapsto [0,1]$. There exists a $\Omega(\min\left\{t'(n), \frac{1}{\varepsilon(n)}\right\})$-pseudorandom generator, with input length is $\Theta(\frac{n^3 \cdot \log \frac{1}{\varepsilon(n)}}{\gamma(n)^2})$, for any t' satisfying $p(n,t'(n)) < t(n)$.

Combining the above proposition and Theorem 5.5.8 we get the main result of this section.

THEOREM 5.5.10
Let $f : \{0,1\}^n \mapsto \{0,1\}^{\ell}(n)$ be a $t(n)$-one-way function and assume that $\log(t(n))$ is polynomially approximable up to a constant ratio. Then there exists a polynomial p such that the following holds for any $\varepsilon : \mathbb{N} \mapsto [0,1]$. There exists an $\Omega(\min\left\{t'(n), \frac{1}{\varepsilon(n)}\right\})$-pseudorandom generator, with input length is $\Theta(\frac{n^7 \cdot \log \frac{1}{\varepsilon(n)}}{(\log t(n))^3})$, for any t' satisfying $p(n, t'(n)) < t(n)$. In particular, by taking $\varepsilon(n) = \frac{1}{t(n)}$ we get a generator with input length $\Theta(n^7)$.

Proof. Let $\gamma(n) = \log(t(n))/3n$. Theorem 5.5.8 guarantees the existence of a polynomial p' for which of a pair (g, b) which is $(t(n), \frac{1}{2}, \gamma(n))$-PEP for any t' satisfying $p'(n, t'(n)) < t(n)$ (in the following we assume wlog that $\gamma(n)$ is polynomially computable, the case where we can only approximate $\log(t(n))$ easily follows along the same lines). Since the value of $\delta(n)$ is fixed (i.e., $\frac{1}{2}$), we do not need the advice α_n. Therefore, it is left to take care of the non-uniform advice β_n (note that the value of $\mathrm{H}(g(W_n))$ is not necessarily efficiently approximable). The overcome this problem we used the following pseudorandom generators "combiner". (Since the combiner we are using was previously used by other applications, e.g., [HILL99, Proposition 4.17] and [Hol06a, Theorem 1], we only give a high-level overview of its construction and proof.). For any $\lambda \in [0,1]$, let G_λ be the generator resulting from applying Proposition 5.5.9 with $\beta_n = \lambda$. Let G'_λ be the result of applying the [GGM86] length-extension method to G_λ such that the output length of G'_λ is $m = \lceil 4n/\gamma(n) \rceil$ times longer than the input length of G_λ. Finally let $G(x_1, \ldots, x_m) = \bigoplus_{i=0}^{m} G'_{\frac{i}{m}}(x_i)$. Clearly G is length expanding, polynomial-time computable and has input length $\Theta(\frac{n^4 \cdot \log \frac{1}{\varepsilon(n)}}{\gamma(n)^3}) = \Theta(\frac{n^7 \cdot \log \frac{1}{\varepsilon(n)}}{(\log t(n))^3})$. By a standard hybrid argument, any algorithm B of running-time $t_B(n)$, which distinguishes between the output of G from the uniform distribution with advantage $\frac{1}{t_B(n)}$, implies the following for any $i \in [m]$. There exist an efficient algorithm M_i^B, with oracle access to B, that distinguishes between the output of $G_{\frac{i}{m}}$ from the uniform distribution with advantage $\Omega(\frac{1}{t_B(n) \cdot \mathrm{poly}(n)})$. We conclude that for large enough p, it holds that $p(n, t_B(n)) > t(n)$. □

5.6 HILL's Pseudo-Entropy Pair

In the following we complete the picture of Section 5.5 by presenting the pseudoentropy pair used in [HILL99].

CONSTRUCTION 5.6.1 (the HILL PEP)
Let $f : \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}$ be a one-way function and let b be any hardcore predicate of f. Let $\mathcal{H}$ be an efficient family of pairwise-independent hash functions from $\{0,1\}^n$ to $\{0,1\}^n$. Let $f_{\mathrm{H}}(x,h,i) = (f(x), h_{i+2\lceil log(n) \rceil}(x), h, i)$ and $b_{\mathrm{H}}(x,h,i) = b(x)$, where $x \in \{0,1\}^n$, $h \in \mathcal{H}$, $i \in [n]$ and $h_k(x)$ stands for the first k bits of $h(x)$.

[HILL99] proves that $(f_{\mathrm{H}}, b_{\mathrm{H}})$ is a $(p + \frac{1}{2n}, \alpha)$-PEP, where $p \stackrel{\mathrm{def}}{=} \Pr_{(x,i) \leftarrow (U_n, [n])}[D_f(f(x)) < i]$ and α is any fraction noticeably smaller than $\frac{1}{n}$. The proof goes by showing that it is hard to

predict the hardcore-bit of an input element $(x,h,i) \in \mathrm{Im}(f_{\mathrm{H}})$ whenever $i \leq D_f(f(x))$. Roughly speaking, the reason is that in such a case, $(h_{i+2\lceil log(n) \rceil}(x),h,i)$ does not contain any noticeable information about x. Thus, it is essentially as hard to predict $b_{\mathrm{H}}(x,h,i) = b(x)$ given $f_{\mathrm{H}}(x,h,i)$ as it is to predict $b(x)$ given $f(x)$. Since the probability that $i = D_f(x)$ is $\frac{1}{n}$, it follows that for any α noticeably smaller than $\frac{1}{n}$ it holds that b_{H} is a $(p+\alpha)$-hard predicate of f_{H}.

On the other hand, by the pairwise independence of $\mathcal{H}$, whenever $i \geq D_f(x)$ there is almost no entropy (i.e., less then $1/2n$) in $b_{\mathrm{H}}(x,h,i)$ given $f_{\mathrm{H}}(x,h,i)$. Thus, the entropy of $b_{\mathrm{H}}(x,h,i)$ given $f_{\mathrm{H}}(x,h,i)$ is not more than $p + \frac{1}{2n}$.

REMARK 5.6.2

Note that the above b_{H} is not only $(t(n), 1-\delta(n)-\alpha(n))$-hardcore of f_{H}, but its hardness comes from the existence of a "hardcore-set" of density $\delta+\alpha(n)$. Where the latter is a subset of the input such that the value of b_{H} is computationally unpredictable over it. This additional property was used by [HILL99] original implementation of pseudorandom generator, but it is not required by the new proof due to [Hol06a]. We note, however, that our PEP presented in Section 5.5 also has such a hardcore set.

Chapter 6

Hardness Amplification of Regular One-way Functions

6.1 Introduction

The existence of one-way functions is essential to almost any task in cryptography (see for example [IL89]) and also sufficient for numerous cryptographic primitives, such as the pseudorandom generators discussed above. In general, for constructions based on one-way functions we use what are called *strong* one-way functions. That is, functions that can only be inverted efficiently with negligible success probability. A more relaxed definition is that of an δ-weak one-way function where $\delta(n)$ is a polynomial fraction. This is a function that no efficient algorithm inverts with probability better than $1-\delta(n)$. This definition is significantly weaker, yet Yao [Yao82] showed how to convert any weak one-way function into a strong one. The new strong one-way function simply consists of many independent copies of the weak function concatenated to each other. The solution of Yao, however, incurs a blow-up factor of at least $\omega(1)/\delta(n)$ to the input length of the strong function, [1] which translates to a significant loss in the security (as in the case of pseudorandom generators).

With this security loss in mind, several works have tried to present an efficient method of amplification from weak to strong. Goldreich et al. [GIL+90] give a solution for one-way permutations that has just a linear blowup in the length of the input. This solution generalizes to *known-regular* one-way functions (regular functions whose image size is efficiently computable), where its input length varies according to the required security. The input length is linear when security is at most $2^{\Omega(\sqrt{n})}$, but deteriorates up to $O(n^2)$ when the required security is higher (e.g., security $2^{O(n)}$).[2] Their construction uses a variant of randomized iterates discussed in Chapter 5, where the randomization is via one random step on an expander graph.

6.1.1 Our Results

We present an alternative efficient hardness amplification for regular one-way functions. Specifically, in Theorem 6.2.1 we show that the m'th randomized iterate of a weak one-way function along

[1] The $\omega(1)$ factor stands for the logarithm of the required security. For example, if the security is $2^{O(n)}$ then this factor is of order n.

[2] Loosely speaking, one can think of the security as the probability of finding an inverse to a random image $f(x)$ simply by choosing a random element in the domain.

with the randomizing hash functions form a strong one-way function (for the right parameter m). Moreover, this holds also for the derandomized version of the randomized iterate (Theorem 6.2.8), giving an almost linear construction. Our construction is arguably simpler and has the following advantages:

1. While the [GIL+90] construction works only for *known* regular weak one-way functions, our amplification works for any regular weak one-way functions (whether its image size is efficiently computable or not).
2. The input length of the resulting strong one-way function is $O(n \log n)$ regardless of the required security. Thus, for some range of the parameters our solution is better than that of [GIL+90] (although it is worse than [GIL+90] for other ranges).

Note that our method may yield an $O(n)$ input construction if bounded-space generators with better parameters become available.

6.1.2 Our Techniques

At the basis of all hardness amplification lies the fact that for any inverting algorithm, a weak one-way function has a set that the algorithm fails upon, called here the ***failing-set*** of this algorithm. The idea is that a large enough number of randomly chosen inputs are bound to hit every such failing-set and thus fail every algorithm. Taking independent random samples works well, but when trying to generate the inputs to f sequentially this rationale fails. The reason is that sequential applications of f are not likely to give random output, and hence are not guaranteed to hit a failing-set. Instead, the natural solution is to use randomized iterations. It might be easy, however, for an inverter to find some choice of randomizing hash functions so that all the iterates are outside of the required failing-set. To overcome this, the randomizing hash functions are also added to the output, and thus the inverter is required to find an inverse that includes the original randomizing hash functions. In the case of permutations it is obvious that outputting the randomizing hash functions is harmless, and thus the k'th randomized iterate of a weak one-way permutation is a strong one-way permutation. The case of regular functions, however, requires our analysis that shows that the randomized iterate of a regular one-way function remains hard to invert when the randomizing hash functions are public. We also note that the proof for regular functions has another subtlety. For permutations the randomized iterate remains a permutation and therefore has only a single inverse. Regular functions, on the other hand, can have many inverses. This comes into play in the proof, when an inverting algorithm might not return the right inverse that is actually needed by the proof.

A major problem with the randomized iterate approach is that choosing fully independent randomizing hash functions requires an input as long as that of Yao's solution (an input of length $O(n \cdot \omega(1)/\delta(n))$). What makes this approach appealing after all, is the derandomization of the hash functions using space-bounded generators, which reduces the input length to only $O(n \log n)$. Note that in this application of the derandomization, it is required that the bounded-space generator not only approximate the collision-probability well, but also maintain the high probability of hitting any failing-set.

We note that there have been several attempts to formulate such a construction, using all of the tools mentioned above. Goldreich et al. [GIL+90] did actually consider following the GKL methodology, but chose a different (though related) approach. Phillips [Phi93] gives a solution with

input length $O(n \log n)$ using bounded-space generators but only for the simple case of permutations (where [GIL+90] has better parameters). Di Crescenzo and Impagliazzo [DI99] give a solution for regular functions, but only in a model where public randomness is available (in the mold of [HL92]). Their solution is based on pairwise-independent hash functions that serve as the public randomness. We are able to combine all of these ingredients into one general result, perhaps due to our simplified proof.

6.2 The Amplification

We start (Section 6.2.1) with a basic construction that archives the required amplification, but is not more efficient than Yao's construction for general one-way functions. We then (Section 6.2.2) apply the same ideas as in Section 5.3.4 to get our efficient amplification.

6.2.1 The Basic Construction

For simplicity we assume that the underlying weak one-way function is length-preserving, where the adaptation to any regular one-way function is the same as in Section 5.3. As a first step, we show that $g(x, \overline{h}) = (f^m(x, \overline{h}), \overline{h})$ is a strong one-way function (for the proper choice of m). The basic intuition is that every iteration of the randomized iterate gives a random element in $Im(f)$ and thus these iterations are bound to hit every significantly large failing-set. The actual proof, however, is more subtle than this.

THEOREM 6.2.1
Let $\delta(n) > 1/\operatorname{poly}(n)$ and let $f : \{0,1\}^n \mapsto \{0,1\}^n$ be a $(t(n), 1-\delta(n))$-one-way function. Let $m = \left\lceil \frac{4n}{\delta(n)} \right\rceil$, and let $\mathcal{H}$ and f^k be as in Definition 5.3.1. We define $g : \{0,1\}^n \times \mathcal{H}^m \mapsto Im(f) \times \mathcal{H}^m$ as $g(x, \overline{h}) = (f^m(x, \overline{h}), \overline{h})$. Then there exists a polynomial p such that g is a $t'(n)$-one-way for any t' satisfying $p(n, t'(n)) < t(n)$.

Proof. We start by showing that f has a large failing-set, and then show that any algorithm that inverts g too well, contradicts the existence of such a set.

DEFINITION 6.2.2 (failing-set)
Let $f : \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}$, let $\varepsilon, \delta : \mathbb{N} \mapsto [0,1]$ and let A be an algorithm trying to invert f. The set $S \subseteq Im(f(\{0,1\}^n)$ is a (ε, δ)-failing-set for A, if $\Pr[f(U_n) \in S] \geq \delta(n)$ and $\Pr[A(y) \in f^{-1}(y)] < \varepsilon(n)$ for every $y \in S$.

LEMMA 6.2.3
Let $\delta(n) > 2^{-n+1}$ and let $f : \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}$ be a $(t(n), 1-\delta(n))$-one-way function. Then there exists a polynomial p such that every algorithm of running at most $t'(n)$ satisfying $p(n, t'(n)) < t(n)$, has a $(1/t'(n), \delta(n)/2)$-failing-set.

Proof. Let A be an algorithm of running-time at most $t_A(n)$ trying to invert f and let $S \subseteq Im(f)$ be the set of elements that A inverts with probability less than $1/t_A(n)$ (i.e., for each $y \in S$ it

holds that $\Pr[A(y) \in f^{-1}(y)] < 1/t_A(n)$). Assume that $\Pr[f(U_n) \in S] < \delta(n)/2$ and consider the algorithm A' for inverting f. On each input ($y \in Im(f)$) the algorithm invokes A for $nt_A(n)$ independent times (with fresh random coins) and returns a preimage of y if at least one of the instances of A finds such a preimage. The running time of A' is $O(n \cdot t_A(n)(t_A(n) + t_f(n)))$, where $t_f(n)$ is the evaluation time of f. Also, on the elements outside S, it is guaranteed that A' fails to invert f with probability at most $(1 - \frac{1}{t_A(n)})^{nt_A(n)} < 2^{-n}$. Thus, overall A' succeeds in inverting f with probability at least $1 - \frac{\delta(n)}{2} - 2^{-n} > 1 - \delta(n)$. it follows, by the properties of f, that the running-time of A' has to be at least $t(n)$. Thus, for the right choice of p we get $p(n, t_A(n)) > t(n)$ as required. □

Let A be an algorithm that runs in time $t_A(n)$ and inverts g with probability $1/t_A(n)$. The next lemma shows that there exists an algorithm M^A, that runs in time $t_M(n) \leq p(n, t_A(n))$ for some polynomial p, yet has no $(1/t_M(n), \delta(n)/2)$-failing-set for f. It therefore implies that $t_A(n)$ must be large; namely, that g is one-way.

LEMMA 6.2.4

Let f and g be as in Theorem 6.2.1. There exists an oracle aided algorithm $M^{(\cdot)}$ and a polynomial p such that the following holds. Let A be an algorithm that runs in time $t_A(n)$ and inverts g with probability $1/t_A(n)$. If algorithm M^A runs in time $t_{M^A}(n) < p(n, t_A(n))$ then for every set S with $\Pr[f(U_n) \in S] \geq \frac{\delta}{2}$, it holds that $\Pr[M^A(f(U_n)) = f^{-1}(U_n) \wedge f(U_n) \in S] \geq \frac{1}{t_{M^A}(n)}$.

Before proving Lemma 6.2.4 we first show how to use it for proving Theorem 6.2.1. The lemma states that for every set S for which $\Pr[f(U_n) \in S] \geq \frac{\delta}{2}$, there exists $y \in S$ such that $\Pr[M^A(y) = f^{-1}(y)] \geq \frac{1}{t_{M^A}(n)}$. Thus, f does not have a $(1/t_M(n), \delta(n)/2)$-failing-set and by Lemma 6.2.3 it must be that $p'(n, t_M(n)) > t(n)$ for some polynomial p'. □

Proof. (of Lemma 6.2.4) For every $i \in [m]$ we define M_i^A as the algorithm that tries to use the i'th iteration in order to invert the function f. Specifically:

ALGORITHM 6.2.5

Algorithm M_i^A for inverting the last-iteration of f^i.

Input: $(y, h_1 \ldots, h_i) \in Im(f) \times \mathcal{H}^i$.

1. Choose uniformly and independently at random $h_{i+1}, \ldots, h_m \in \mathcal{H}$ and let $w = f^{m-i}(y, h_{i+1}, \ldots, h_m)$.
2. Apply $A(w, h_1, \ldots, h_m)$ to get $(x, h'_1, \ldots, h'_m)$.
3. Return $h_i(f^{i-1}(x, h_1, \ldots, h_{i-1}))$.

The algorithm M^A then tries using each of the possible iterations for inverting f. Formally:

ALGORITHM 6.2.6

Algorithm M^A for inverting f.

Input: $y \in Im(f)$.

1. Choose a random $h_1, \ldots, h_m \in \mathcal{H}$.
2. For each $i \in [m]$ set $x_i = M_i^A(y, h_1, \ldots, h_i)$.
3. If there exists an i such that $f(x_i) = y$ output x_i, otherwise abort.

For clarity we omit the value n whenever it is clear from the context (i.e., we write δ rather than $\delta(n)$). To prove Lemma 6.2.4 we need to show that M^A succeeds in inverting f over every large enough set S. This follows by first showing (in Claim 6.2.7) that there exists at least one iteration $i \in [m]$ that M_i^A succeeds on. Namely, M_i^A succeeds in inverting the i'th randomized iteration of f. This is then combined with the fact that inverting the i'th randomized iterate can be used to invert f (as shown in section 5.3, Lemma 5.3.5).

CLAIM 6.2.7

For any subset $S \subseteq Im(f)$ of density at least $\delta/2$ there exists an index $j \in [m]$ for which $\Pr[f(M_j^A(f^j(U_n, H^j), H^j)) = f^j(U_n, H^j) \wedge f^j(U_n, H^j) \in S] \in \Omega(\frac{1}{t_A^2 \cdot m^2})$.

We defer for now the proof of Claim 6.2.7 and first use it to prove Lemma 6.2.4. For ease of notation, define $\alpha_{m,A} = max_{j \in [m]}\{\Pr[f(M_j^A(f^j(U_n, H^j), H^j)) = f^j(U_n, H^j) \wedge f^j(U_n, H^j) \in S]\}$. That is, the claim above states that $\alpha_{m,A} \in \Omega(\frac{1}{t_A^2 \cdot m^2})$. Let $S \subseteq Im(f)$ be any subset of density at least $\delta/2$. For $i \in [m]$ let $L_i \stackrel{\text{def}}{=} \left\{(y, \overline{h}) \in S \times \mathcal{H}^i : \Pr[f(M_i^A(y, \overline{h})) = y] \geq \alpha_{m,A}/2\right\}$, where the probability is over the random coins of M_i^A (namely, L_i is the set of elements that M_i^A inverts their last iteration w.h.p.). Note that for every $i \in [m]$ it holds that

$$\begin{aligned}
& \Pr[f(M_i^A(f^i(U_n, H^i), H^i)) = f^i(U_n, H^i) \wedge f^i(U_n, H^i) \in S] \\
= \; & \Pr[f(M_i^A(f^i(U_n, H^i), H^i)) = f^i(U_n, H^i) \wedge f^i(U_n, H^i) \in S \wedge (f^i(U_n, H^i), H^i) \in L_i] \\
+ \; & \Pr[f(M_i^A(f^i(U_n, H^i), H^i)) = f^i(U_n, H^i) \wedge f^i(U_n, H^i) \in S \wedge (f^i(U_n, H^i), H^i) \notin L_i] \\
\leq \; & \Pr[(f^i(U_n, H^i), H^i) \in L_i] + \alpha_{m,A}/2 \ .
\end{aligned}$$

Thus, there exists $j \in [m]$ such that $\Pr[(f^j(U_n, H^j), H^j) \in L_j] \geq \alpha_{m,A}/2$. By Lemma 5.3.5 (letting $T = L_j$) it follows that $\Pr[(f(U_n), H^j) \in L_j] \in \Omega(\alpha_{m,A}^2/m)$. Since M_j^A inverts the last-iteration of each of the elements in L_j with probability $\alpha_{m,A}/2$, it follows that

$$\begin{aligned}
& \Pr[M^A(f(U_n)) = f^{-1}(U_n) \wedge f(U_n) \in S] \\
\geq \; & \Pr[f(M_j^A(f(U_n), H^j)) = f(U_n) \wedge f(U_n) \in S] \in \Omega(\alpha_{m,A}^3/m) = \Omega(\frac{1}{t_A^6 \cdot m^7}) \ .
\end{aligned}$$

Finally, since each invocation of M_i^A involves m invocations of f and a single invocation of A,we have that $t_M(n) < p(n, t_A(n))$ for the right choice of p. □

Proof. (of Claim 6.2.7) Through the rest of this section we allow ourselves to view g and f^m as functions over $Im(f) \times \mathcal{H}^m$ rather than over $\{0,1\}^n \times \mathcal{H}^m$, where for $y \in Im(f)$ we let $f^m(y, \overline{h}) = f^m(x, \overline{h})$ for some $x \in f^{-1}(y)$.[3] For any $(w, \overline{h}) \in Im(g)$ let $B_{w,\overline{h}} \stackrel{\text{def}}{=} \left\{y \in Im(f) : f^m(y, \overline{h}) = w\right\}$.

[3]Note that the above is well defined since for any $x, x' \in f^{-1}(y)$ it holds that $f^m(x, \overline{h}) = f^m(x', \overline{h})$).

It readily follows from the proof of Claim 5.3.6 that for any $y \neq y' \in Im(f)$ it holds that $\Pr[f^m(y, H^m) = f^m(y', H^m)] \leq \frac{m}{|Im(f)|}$. Thus, for any $y \in Im(f)$ it holds that $\mathsf{Ex}[|B_{g(y,H^m)}|] = 1 + \sum_{y' \neq y \in Im(f)} \mathsf{Ex}[f^m(y', H^m) = f^m(y, H^m)] < m + 1$. By Markov's inequality $\Pr[|B_{g(x,\overline{h})}| > 4m \cdot t_A] < \frac{1}{2 \cdot t_A}$ and therefore

$$\Pr[A(g(U_n, H^m)) \in g^{-1}(g(U_n, H^m)) \wedge |B_{g(U_n,H^m)}| \leq 4m \cdot t_A] > \frac{1}{2 \cdot t_A} \tag{6.1}$$

Now let $S \subseteq Im(f)$ be a set of density $\delta/2$. By the pairwise independence of $\mathcal{H}$ (actually one-wise independence suffices for this part), we have that for every $y \in Im(f)$, $i \in [m]$ and $\overline{h} \in \mathcal{H}^{i-1}$, it holds that $\Pr[f^i(y, (\overline{h}, H)) \in S] \geq \delta/2$. Hence, for any $y \in Im(f)$ it holds that $\Pr[\exists i \in [m] \ : \ f^i(y, H^m) \in S] > 1 - 2^{-2n}$ (recall that $m = \lceil \frac{4n}{\delta} \rceil$). By a union bound,

$$\begin{aligned} & \Pr[\forall y \in B_{g(U_n,H^m)} \ \exists i \in [m] \ : \ f^i(y, H^m) \in S] \\ > \ & 1 - |B_{g(U_n,H^m)}| \cdot 2^{-2n} \geq 1 - 2^{-n} \end{aligned} \tag{6.2}$$

Combining (6.1) and (6.2) yields that,

$$\Pr\left[\begin{array}{c} A(g(U_n, H^m)) \in g^{-1}(g(U_n, H^m)) \ \wedge \ |B_{g(U_n,H^m)}| \leq 4m \cdot t_A \\ \wedge \forall y \in B_{g(U_n,H^m)} \ \ \exists i \in [m] \ : \ f^i(y, H^m) \in S \end{array}\right] > \frac{1}{2 \cdot t_A} - 2^{-n} > \frac{1}{4 \cdot t_A} \ ,$$

where the last inequality follows since wlog $t_A < 2^{n/2}$. Since for any $(x', \overline{h}') \in g^{-1}(z)$ it holds that then $f(x') \in B_z$, we have that

$$\Pr\left[\begin{array}{c} A(g(U_n, H^m)) \in g^{-1}(g(U_n, H^m)) \wedge |B_{g(U_n,H^m)}| \leq 4m \cdot t_A \\ \wedge \exists i \in [m] \ : \ f^i(A(g(U_n, H^m))) \in S \end{array}\right] > \frac{1}{4 \cdot t_A} \ .$$

Thus, by an averaging argument there exists an index $j \in [m]$ such that,

$$\Pr[A(g(U_n, H^m)) \in g^{-1}(g(U_n, H^m)) \wedge |B_{g(U_n,H^m)}| \leq 4m \cdot t_A \wedge f^j(A(g(U_n, H^m))) \in S] > \frac{1}{4 \cdot t_A} \ .$$

For $z \in Im(g)$ let $A(z)_1$ be the first part of $A(z) = (x, \overline{h})$ (i.e., x). The regularity of f yields that

$$\Pr[f(A(g(U_n, H^m))_1) = f(U_n) \wedge f^j(A(g(U_n, H^m))) \in S] > \frac{m}{4 \cdot t_A} \cdot \frac{m}{4 \cdot t_A} = \frac{m^2}{16 \cdot t_A^2} \ ,$$

and we conclude that

$$\begin{aligned} & \Pr[f(M_j^A(f^j(U_n, H^j), H^j)) = f^j(U_n, H^j) \wedge f^j(U_n, H^j) \in S] \\ = \ & Pr[f(A(g(U_n, H^m))_1) = f(U_n) \wedge f^j(A(g(U_n, H^m))) \in S] \\ > \ & \frac{m^2}{16 \cdot t_A^2} \ . \end{aligned}$$

□

6.2.2 An Almost-Linear-Input Construction

In this section we derandomize the randomized iterate used in Section 6.2.1 to get a (strong) one-way function with input length $O(n \log n)$. We use the bounded-space generator of either [Nis92] or [INW94] (see Section 5.2.4).

THEOREM 6.2.8

Let f, m, $\mathcal{H}$ and f^m be as in Theorem 6.2.1. Let $v(\mathcal{H})$ be the description length of $h \in H$ and let $BSG : \{0,1\}^{\tilde{n} \in O(n \log n)} \mapsto \{0,1\}^{mv(\mathcal{H})}$ be a bounded-space generator that 2^{-2n}-fools every $(2n, m, v(\mathcal{H}))$-LBP. Define $g' : \{0,1\}^n \times \{0,1\}^{\tilde{n}} \mapsto \{0,1\}^n \times \{0,1\}^{\tilde{n}}$ as $g(x, \tilde{h}) = (f^m(x, BSG(\tilde{h})), \tilde{h})$, where $x \in \{0,1\}^n$ and $\tilde{h} \in \{0,1\}^{\tilde{n}}$.

Then there exists a polynomial p such that g' is a $(t'(n), \frac{1}{t'(n)})$-one-way for any t' satisfying $p(n, t'(n)) < t(n)$.

Proof idea: The proof of the derandomized version follows the proof of Theorem 6.2.1. In the proof we used the following properties of the family $\mathcal{H}$.

Collision-probability - for any $y \neq y' \in Im(f)$ and $i \in \{0, \ldots, m\}$

$$\Pr[f^i(y, H^i) = f^i(y', H^i)] = \frac{i}{|Im(f)|}$$

Hitting - for all $y \in Im(f)$ and $S \subseteq Im(f)$ of density $\delta/2$

$$\Pr[\exists i \in [m] \ : \ f^m(y, H^m) \in S] > 1 - 2^{-2n}$$

Note that the above two properties can be verified by an $(n, m, v(\mathcal{H}))$-LBP. Since, BSG 2^{-2n}-fools such LBP's, the above properties hold with deviation at most 2^{-2n} w.r.t. to $\overline{h} = BSG(\tilde{n})$. Going through the proof of Theorem 6.2.1, it is not hard to verify that the proof remains valid also when the above deviations are taking into account. (See the proof of Theorem 5.3.11 for a more detailed proof on a similar derandomization).

Publications and Statement of Originality

Most of the results presented in this dissertation have been published as follows.

1. Iftach Haitner and Omer Reingold. Statistically-hiding commitment from any one-way function. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*. ACM Press, 2007.

   Presented in Chapter 3, parts of the write-up are taken from [HNO+07].

2. Iftach Haitner and Omer Reingold. A new interactive hashing theorem. In *Proceedings of the 18th Annual IEEE Conference on Computational Complexity*, 2007.

   Presented in Chapter 4.

3. Iftach Haitner, Danny Harnik, and Omer Reingold. Efficient pseudorandom generators from exponentially hard one-way functions. In *Automata, Languages and Programming, 24th International Colloquium, ICALP*, 2006.

   Presented in Chapters 5 and 6.

4. Iftach Haitner, Danny Harnik, and Omer Reingold. On the power of the randomized iterate. In *Advances in Cryptology – CRYPTO 2006*, 2006.

   Presented in Chapter 5.

The dissertation contains additional results beyond the aforementioned publications.

All results presented in this dissertation are, where not stated otherwise, original research. Most of this research was done in collaboration with other investigators, as reflected in the aforementioned publications. My personal contribution to each of these publications was substantial at the conceptual, technical and editorial levels.

Bibliography

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. *SIAM Journal on Computing*, 36, 2006.

[BBR88] Charles H. Bennett, Gilles Brassard, and Jean-Marc Robert. Privacy amplification by public discussion. *SIAM Journal on Computing*, 17(2):210–229, 1988.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.

[BCY91] Gilles Brassard, Claude Crépeau, and Moti Yung. Constant-round perfect zero-knowledge computationally convincing protocols. *Theoretical Computer Science*, 84(1):23–52, 1991.

[BKK90] Joan F. Boyar, Stuart A. Kurtz, and Mark W. Krentel. A discrete logarithm implementation of perfect zero-knowledge blobs. *Journal of Cryptology*, 2(2):63–76, 1990.

[BM82] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo random bits. pages 112–117, 1982.

[CCM98] Christian Cachin, Claude Crépeau, and Julien Marcil. Oblivious transfer with a memory-bounded receiver. pages 493–502, 1998.

[CDG87] David Chaum, Ivan Damgård, and Jeroen van de Graaf. Multiparty computations ensuring privacy of each party's input and correctness of the result. In *Advances in Cryptology – CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 87–119. Springer, 1987.

[CS06] Claude Crépeau and George Savvides. Optimal reductions between oblivious transfers using interactive hashing. In *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 201–221. Springer, 2006.

[CW79] J. Lawrence Carter and Mark N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, April 1979.

[DHRS04] Yan Zong Ding, Danny Harnik, Alon Rosen, and Ronen Shaltiel. Constant-round oblivious transfer in the bounded storage model. In *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004*, pages 446–472, 2004.

[DI99] Giovanni Di Crescenzo and Russell Impagliazzo. Security-preserving hardness-amplification for any regular one-way function. In *31st STOC*, pages 169–178, 1999.

[DPP98] Ivan B. Damgård, Torben P. Pedersen, and Birgit Pfitzmann. Statistical secrecy and multibit commitments. *IEEE Transactions on Information Theory*, 44(3):1143–1151, 1998.

[GGKT05] Rosario Gennaro, Yael Gertner, Jonathan Katz, and Luca Trevisan. Bounds on the efficiency of generic cryptographic constructions. *SIAM Journal on Computing*, 35(1):217–246, 2005.

[GGL98] Oded Goldreich, Shafii Goldwasser, and Nathan Linial. Fault-tolerant computation in the full information model. *SIAM Journal on Computing*, 27, 1998.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986.

[GIL+90] Oded Goldreich, Russell Impagliazzo, Leonid A. Levin, Ramarathnam Venkatesan, and David Zuckerman. Security preserving amplification of hardness. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 318–326, 1990.

[GK96] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–190, 1996.

[GKL93] Oded Goldreich, Hugo Krawczyk, and Michael Luby. On the existence of pseudorandom generators. *SIAM Journal on Computing*, 22(6):1163–1175, 1993.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 25–32, 1989.

[GMR88] 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, 1988. Preliminary version in *FOCS'84*.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. pages 218–229, 1987.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(1):691–729, 1991. Preliminary version in *FOCS'86*.

[Gol00] Oded Goldreich. On security preserving reductions — revised terminology. Technical Report 2000/001, Cryptology ePrint Archive, 2000.

[Gol01a] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.

[Gol01b] Oded Goldreich. Randomized methods in computation - lecture notes. 2001.

[Hås90] Johan Håstad. Pseudo-random generators under uniform assumptions. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 387–394, 1990.

[HHK+05] Iftach Haitner, Omer Horvitz, Jonathan Katz, Chiu-Yuen Koo, Ruggero Morselli, and Ronen Shaltiel. Reducing complexity assumptions for statistically-hiding commitment. In *Advances in Cryptology – EUROCRYPT 2005*, pages 58–77, 2005. See also preliminary draft of full version, www.wisdom.weizmann.ac.il/~iftachh/papers/SCfromRegularOWF.pdf.

[HHR06] Iftach Haitner, Danny Harnik, and Omer Reingold. On the power of the randomized iterate. In *Advances in Cryptology – CRYPTO 2006*, 2006.

[HHRS07] Iftach Haitner, Jonathan J. Hoch, Omer Reingold, and Gil Segev. Finding collisions in interactive protocols – A tight lower bound on the round complexity of statistically-hiding commitments. In *Proceedings of the 47th Annual Symposium on Foundations of Computer Science (FOCS)*, 2007.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. Preliminary versions in *STOC'89* and *STOC'90*.

[HK05] Omer Horvitz and Jonathan Katz. Bounds on the efficiency of "black-box" commitment schemes. In *ICALP '05*, pages 128–139, 2005.

[HL92] Amir Herzberg and Michael Luby. Pubic randomness in cryptography. In *Advances in Cryptology – CRYPTO '92*, volume 740, pages 421–432. Springer, 1992.

[HNO+07] Iftach Haitner, Minh-Huyen Nguyen, Shien Jin Ong, Omer Reingold, and Salil Vadhan. Statistically-hiding commitments and statistical zero-knowledge arguments from any one-way function. Unpublished manuscript, November 2007.

[Hol05] Thomas Holenstein. Key agreement from weak bit agreement. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 664–673, 2005.

[Hol06a] Thomas Holenstein. Pseudorandom generators from one-way functions: A simple construction for any hardness. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006*, 2006.

[Hol06b] Thomas Holenstein. Strengthening key agreement using hard-core sets - PhD thesis, 2006.

[HR07] Iftach Haitner and Omer Reingold. Statistically-hiding commitment from any one-way function. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*. ACM Press, 2007.

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 230–235, 1989.

[ILL89a] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 12–24, 1989.

[ILL89b] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 12–24. ACM Press, 1989.

[INW94] Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 356–364, 1994.

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 44–61. ACM Press, 1989.

[IZ89] Russell Impagliazzo and David Zuckerman. How to recycle random bits. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 248–253, 1989.

[KK05] Jonathan Katz and Chiu-Yuen Koo. On constructing universal one-way hash functions from arbitrary one-way functions. Technical Report 2005/328, Cryptology ePrint Archive, 2005.

[KS06] Takeshi Koshiba and Yoshiharu Seri. Round-efficient one-way permutation based perfectly concealing bit commitment scheme. ECCC, TR06-093, 2006.

[KST99] Jeong Han Kim, Daniel R. Simon, and Prasad Tetali. Limits on the efficiency of one-way permutation-based hash functions. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 535–542, 1999.

[Lev87] Leonid A. Levin. One-way functions and pseudorandom generators. *Combinatorica*, 7:357–363, 1987.

[Lin03] Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. *JCRYPTOLOGY*, 2003.

[LR88] Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, 1988.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991. Preliminary version in *CRYPTO'89*.

[Nis92] Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

[NN93] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM Journal on Computing*, 1993.

[NOV06] Minh-Huyen Nguyen, Shien Jin Ong, and Salil Vadhan. Statistical zero-knowledge arguments for NP from any one-way function. In *Proceedings of the 47th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 3–14, 2006.

[NOVY98] Moni Naor, Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Perfect zero-knowledge arguments for NP using any one-way permutation. *Journal of Cryptology*, 11(2):87–108, 1998. Preliminary version in *CRYPTO'92*.

[NV06] Minh-Huyen Nguyen and Salil Vadhan. Zero knowledge with efficient provers. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 287–295. ACM Press, 2006.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 33–43. ACM Press, 1989.

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996.

[OVY92] R. Ostrovsky, R. Venkatesan, and M. Yung. Secure commitment against A powerful adversary. In *9th Annual Symposium on Theoretical Aspects of Computer Science*, volume 577 of *lncs*, pages 439–448, Cachan, France, 13–15 February 1992. Springer.

[OVY93a] Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Fair games against an all-powerful adversary. *AMS DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 155–169, 1993. Preliminary version in *SEQUENCES'91*.

[OVY93b] Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Interactive hashing simplifies zero-knowledge protocol design. In *Advances in Cryptology – EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 267–273. Springer, 1993.

[OW93] Rafail Ostrovsky and Avi Wigderson. One-way functions are essential for non-trivial zero-knowledge. In *Proceedings of the 2nd Israel Symposium on Theory of Computing Systems*, pages 3–17. IEEE Computer Society, 1993.

[Ped91] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology – CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1991.

[Phi93] Security preserving hardness amplification using PRGs for bounded space. Preliminary Report, Unpublished, 1993.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 387–394, 1990.

[RTV04] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2004.

[Sim98] Daniel Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In *Advances in Cryptology – EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 334–345. Springer, 1998.

[Top01] Flemming Topsøe. Bounds for entropy and divergence for distributions over a two-element set. *Journal of Inequalities in Pure and Applid Mathematics*, 2001.

[WC81] Mark N. Wegman and J. Lawrence Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 1981.

[Wee07] Hoeteck Wee. One-way permutations, interactive hashing and statistically hiding commitments. pages 419–433, 2007.

[Yao82] Andrew C. Yao. Theory and applications of trapdoor functions. pages 80–91, 1982.