Theory and Application of Extractable Functions

A Dissertation

Presented to the Faculty of the Graduate School

of

Yale University

in Candidacy for the Degree of

Doctor of Philosophy

by

Ramzi Ronny Dakdouk

Dissertation Director: Joan Feigenbaum

December 2009

To my darling Silva

Abstract

Theory and Application of Extractable Functions

Ramzi Ronny Dakdouk

2009

We propose a new cryptographic primitive, called **extractable functions**. An extractable function guarantees any machine that manages to output a point in the range of this function knows a corresponding preimage.

We capture "knowledge of preimage" by way of algorithmic extraction. We formulate two main variants of extractability, namely noninteractive and interactive. The noninteractive variant can be regarded as a generalization from specific knowledge assumptions to a notion that is formulated in general computational terms. Indeed, we show how to realize it under several different assumptions. On the other hand, interactive extraction can be realized from certain perfectly one-way (POW) functions or verifiable secret-sharing (VSS) schemes.

We then initiate a more general study of extractable function aimed at understanding the concept of extractability in of itself. In particular we demonstrate that a weak notion of extraction implies a strong one, and make rigorous the intuition that extraction and obfuscation are complementary notions.

We demonstrate the usefulness of the new primitive in two quite different settings. First, we show how extractable functions can be used to capture, in the standard model, the "knowledge of queries" property that is so useful in the Random Oracle (RO) model. Specifically, we show how to convert a class of CCA2-secure encryption schemes in the RO model to concrete ones by simply replacing the Random Oracle with an extractable function, without much change in the logic of the original proof. Second, we show how extractable functions can be used to construct 3-round ZK arguments using weaker knowledge assumptions than previous results due to Hada and Tanaka (Crypto 1998) and Lepinski (M.S. Thesis, 2004). This also opens the door for constructing 3-round ZK arguments based on other assumptions.

Finally, we exploit techniques used in constructing extractable functions to obfuscate point functions with multibit output. A point function with multibit output returns a fixed string on a single input point and zero everywhere else. Obfuscation of such functions has a useful application as a strong form of symmetric encryption where security holds without any assumption on the distribution of the secret key. We provide a construction that obfuscates these functions. This construction is generic in the sense that it can use any POW function or obfuscator for point functions.

Acknowledgements

I would like to extend my heartfelt gratitude to many people without whom my graduate career would not have been possible.

First and foremost, I would like to thank my advisor, Joan Feigenbaum. From the very beginning, Joan encouraged me to explore different problems that are interesting to me, even though her broader perception of these problems may have at times been different. Joan was extremely helpful through her insightful perception of computer science in general, and old and new subfields in particular. Throughout my graduate career, Joan was constantly encouraging me, and introducing me to researchers, scientists, and to new academic events and programs. Her care transcends my graduate research to include future goals and opportunities. In short, I feel lucky for having Joan as a mentor.

My thanks are due to Ran Canetti who was like a second advisor to me. Ran was always available even when he was busy with deadlines and other commitments. He encouraged me even when my views were less than optimistic. His optimism, great ideas, and enlightening discussions made this work a reality. He was instrumental in helping me abstract away from specific results and understanding them in a broader context.

I am also grateful to Richard Yang and the network group at Yale (specifically, Haiyong Xie and Hao Wang) whom I had the privilege to work with in Summer 2005. Their daily meetings, enthusiasm, and dedication was a precious source of knowledge and motivation for me.

This thesis is the result of joint work with Ran. His contributions are instrumental in shaping this dissertation.

Faculty members of the computer science department were always there with their support, suggestions, and motivation. Notably, I would like to thank Dana Angluin,

James Aspnes, and Michael Fischer.

I also would like to thank the staff of the department, especially, Linda Dobb, Judi Paige, and Judi Smith, for their continuous help in resolving administrative matters even the most formidable ones.

I am also thankful to my colleagues and friends in the department, for making my stay at Yale so pleasant and enjoyable. Specifically, I wish to thank (in no particular order) Alex Vaynberg, Aaron Johnson, Leonor Becerra-Bonache, Nikhil Srivastava, Lev Reyzin, Eli Kim, Edo Liberty, Felipe Saint-Jean, Fred Shic, Pradipta Mitra, Kevin Chang, Aleksandr Yampolskiy, Yinghua Wu, Yitong Yin, Jianye Lu, Nick Ruozzi, Bing Wang and Haluk Tunali for the many pleasant moments going to the gym, lunch, dinner, or even coffee.

Finally, I would like to thank my parents, Ghazi and Randa, my lovely fiancée Silva, and the rest of my family. They gave meaning and life to this work.

This dissertation is funded by NSF grant 0331548.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"Knowledge is recollection"

-Socrates

Even the tripartite theory [Plab], the most common definition of knowledge in epistemology, lends itself to refutation. The tripartite definition of knowledge requires three elements for possession of knowledge. The first element is belief. For one can not know something unless s/he believes in it even if it is true and justifiable. For instance, this thesis may be wholly true and convincing. Yet, if the reader does not believe it, the reader can not claim, according to this definition, knowledge of it. The second element is truth. That is, it is not possible to claim knowledge of something that is not true even if we believe that it is true. For example, in spite of the strong belief of most medieval Europe in the flatness of the earth, we must concede that this belief is in fact unknown. The final element for knowledge is proper justification, i.e., one must provide a convincing argument for the knowledge of something. For instance, guessing the outcome of an experiment is not sufficient for a claim of knowledge, even if we truly believe in our luck and the guess turns out to be correct.

One of the main refutations of the tripartite theory is the Gettier case [Get63]. Two students, Mark and Sam, took an exam. Mark is a straight-A student. He is smart, hard working, and attends all of his classes. Mark wrote throughout the duration of the exam. On the other hand, Sam is lazy, consistently fails, and misses his classes due to illness. Also, Sam scribbled a few lines during the test and then left. Mark said that

he did well on the exam while Sam said that he did not even understand the question. Reflecting on the exam and on a book he read recently, Sam believes that the student with the highest grade shares the same first name with the author of "The Adventures of Huckleberry Finn". Obviously, Sam has proper justification for his belief. Furthermore, this belief is true: the student with the highest grade does in fact have the same first name as the author of this book. However, Sam did not know this. It turns out that Mark did not understand the question and missed its point entirely. On the other hand, Sam understood the question and managed, in writing a few lines, to get a passing grade. Therefore, Sam got the higher grade. Moreover, the name of the author of "The Adventures of Huckleberry Finn" is Samuel Clemens, even though he writes under the pseudonyme of Mark Twain. However, *Sam does not know this.* So, even though Sam has a justified true belief, he does not have knowledge. The problem, pointed out by Socrates [Plab], seems to be in identifying what constitutes proper justification. For instance, is the evidence that Sam has sufficient for a convincing argument?

Even though the question of defining knowledge itself remains, as argued above, open and debatable, we do not attempt to address this question. Instead, we address the question of *communicating* knowledge. In this context, Socrates' definition of knowledge as recollection [Plaa, Pha] is more relevant. Knowledge can be transferred via several means but symbolic representation, i.e., writing, is more relevant computationally. An initial attempt to computational knowledge is to ask a machine to communicate knowledge of something by writing a representation of this thing. However, a machine (or algorithm) is designed to do a specific task and may not be capable of answering even "easy" questions such as "write the square-root of your output". Thus, computational knowledge is taken to mean a machine knows something if there is another machine, similar to the first one, that can communicate, via symbolic representation, knowledge of the thing in question. More succinctly, a machine knows x if there is another machine that has the same environment as the first machine and outputs x. Such a machine is called an **extractor** and computational knowledge is referred to in this thesis as **extractability** or **extraction**.

Extractability plays a central role in cryptographic protocol design and analysis. In its basic form, it relates to two-party protocols where one of the parties (a "prover") has secret input, and tries to convince the other party (a "verifier") that it holds the secret.

The idea is to argue that if the verifier accepts the interaction, then the prover indeed knows the secret. More concretely, extractability makes the following requirement: Given access to the internals of *any* (potentially malicious) prover, it is possible to explicitly and efficiently compute the secret value as long as the verifier accepts an interaction. (Many variants of this notion exist, of course. See e.g. [Gol01].)

In this thesis, we extend the concept of extractability to the more basic setting of computing a function. Here the task of "convincing a verifier" is replaced by "outputting a value in the range of the function". More specifically, any machine that generates a point in the range knows a corresponding preimage in the sense that a preimage is efficiently recoverable given the internal state of the machine. Such functions are called **extractable functions**.

1.1 Our Work

In a single sentence, the goal of this thesis is to introduce, formalize, construct, characterize, and apply extractable functions.

1.1.1 Informal Notion

An efficiently-computable function has an efficient algorithm with the same input/output behavior. In other words, this algorithm takes an input x and returns $f(x)$, where $f(x)$ is the output of the function, f, on point x. One (obvious) statement one can make about this algorithm is that it knows x: clearly, there is a straightforward extractor that outputs its input. However, there may be other algorithms that return points in the range of f. For instance, consider a permutation, π. The standard algorithm for computing π, takes an input x, applies π to x, and returns $\pi(x)$. Different algorithms can also return a point in the range, e.g., by sampling uniformly from this range. Can the same claim be made about such algorithms? In other words, do these machines know a preimage of their output? It turns out that the answer to this question is positive for some functions (extractable functions) and negative for others. The most straightforward function that yields a positive answer to this question is the identity function. On the other hand, one-way permutations do not admit such a property because the algorithm that outputs a random point (as mentioned above) does not know a preimage. This lack

of knowledge is implied by one-wayness: one-wayness prohibits inverting a random point in the range of the permutation.

Crucially, extractable functions require *every* machine that outputs an image to know a preimage. Compare this statement with the fact that every computable function has at least some machines which know a preimage of their output. As an analogy, consider two types of houses. Both types have impenetrable doors except with appropriate keys. However, the first type has breakable windows while the second type has no windows at all. We assign a group of people a rewarding task. The task is to search for a key to the door, then open the door and enter the house. The reward is kept in the house. So, the first person to enter the house gets the reward. If the house is of the first type, it is conceivable that some people avoid the search process, break through the window, and grab the reward. So, it is *not* possible to claim that whoever has the reward has the key. On the other hand, if the house is of the second type, this claim is true. Extractable functions correspond to the second type of houses.

A cryptographic interpretation of this notion is due. One classification of cryptographic players divides them into honest, honest-but-curious, and malicious. The second type follows the prescribed protocol but may compute something extra on the side, while the last type can deviate from the protocol in an arbitrary way. In the context of computing a function, the honest protocol is the standard algorithm for computing the function, i.e., take an input x, apply f on x, and return $f(x)$. Effectively, an extractable function means any malicious adversary is "equivalent" to some honest-but-curious one. An honest-but-curious adversary can simulate the malicious adversary on the side, then run the extractor to recover a preimage, x, and finally compute and output $f(x)$ according to the prescribed protocol. Therefore, extractable functions collapse the classification hierarchy into two: honest and honest-but-curious parties. This fact reduces the task of cryptographers to proving security against honest-but-curious adversaries only.

Useful cryptographic applications require extractable functions to satisfy computational hardness properties. By itself, an extractable function, such as the identity function, has little cryptographic value. Augment it with a hardness property, such as one-wayness [DH76], and an extractable function lends itself to cryptographic applications, as exemplified in this thesis. This duality is crucial and worth highlighting. The output of some machines (typically, honest machines) is hard to invert *from the outside*.

On the other hand, the output of *all* machines is easy to recover *from the inside*. This asymmetry of knowledge between different perspectives is exactly what makes extractable functions succeed where other primitives have failed, such as in 3-round zero-knowledge (see Chapter 6).

1.1.2 Formalization

Defining extractable functions turns out to be a significantly tricky task. A common approach to defining primitives is to examine properties needed by an application and then manifest these properties in a realizable definition. One problem with this approach in this context is that the straightforward manifestation is not realizable. Extraction with *arbitrary* dependent auxiliary information [GK05] contradicts hardness properties, even one-wayness (see Section 3.2). Therefore, we present a series of carefully-crafted definitions that take into account different parameters.

The first parameter is whether extraction is required for a single fixed function or a function chosen randomly from a family. The second parameter, mentioned in the previous paragraph, is the absence or presence of auxiliary information. Auxiliary information can be dependent or independent of the function. The distinction between dependent and independent auxiliary information is relevant only when a function is chosen randomly from a family. A fixed function does not permit distinction between these two types of auxiliary information because it is not possible to prevent independent auxiliary information from depending on this function. Consequently, it is not possible for a single one-way function to be extractable with arbitrary auxiliary information (dependent or independent). Jumping ahead, the best constructions satisfy extraction with independent auxiliary information against a function chosen randomly from a family.

So far, we have discussed extractability in the **noninteractive model**. In this model, extraction means every machine that outputs a *single* image knows a corresponding preimage. One of the main issues with this notion is that known constructions are based on nonstandard assumptions that embody some knowledge "flavor". While realizing this notion from weaker assumptions remains open, we investigate a weaker notion, namely **interactive extraction**.

In the **interactive model**, we relax extraction to mean every machine that returns *"many"* images knows a preimage common to all of them if one exists. For this notion

to be different from noninteractive extraction, there have to be many *different* images of a single input. We can realize this requirement through probabilistic functions, i.e., functions that take two strings x and r as input, where x is labeled the input, and r is labeled the random coins. In this model, $f(x,r)$ is different from $f(x,r')$ if $r \neq r'$. How many images should a machine return? Roughly a polynomial fraction of all images of a single input! Ofcourse, it is impossible to write down in polynomial time these many images if there is an exponential number of them (e.g., if r has length n). To resolve this issue, we introduce an interactive model. In this model, a machine engages in a 3-round Arthur-Merlin protocol with an external agent [Bab85]. This machine sends a single image in the first round. Then, it receives a challenge from the agent. This challenge is in the form of random coins for f. Finally, it responds with new images of the same input using the challenge as random coins for f. Jumping ahead, we use weaker assumptions to realize the strongest notion of interactive extraction, i.e., extraction for a single function with auxiliary information.

1.1.3 Constructions

Constructions with noninteractive extraction. These constructions satisfy a knowledge property and a computational hardness property. The knowledge property is extraction with independent auxiliary information for a randomly-chosen function. The hardness property can be one-wayness, pseudorandomness [Yao82], or perfect one-wayness [Can97]. With the exception of the last one, these constructions are based on nonstandard assumptions with a knowledge "flavor", e.g., the knowledge of exponent assumption [Dam92, HT98] (see also Assumption 3.3.1). The final construction is based on a strong notion of noninteractive zero-knowledge (NIZK) proofs of knowledge [SCP00]. These constructions are presented in more detail in Chapter 3.

Constructions with interactive extraction. These constructions also satisfy a knowledge property and a hardness property. However, the knowledge property is stronger, specifically, extraction with auxiliary information for any function. Moreover, these constructions are based on weaker assumptions of computational hardness nature such as perfect one-wayness or verifiable secret-sharing schemes [CGMA85, Fel87].

The main construction is a transformation from a perfectly one-way (POW) function (with auxiliary information) to extractable POW function. At a high level, this trans-

formation introduces some structure to the output of the function, so that it is easy to recover a preimage from two "related" images. Ofcourse, the issue is to insure that these two related images rarely appear in the 3-round game described above. On the other hand, an extractor can rewind the machine multiple times so that the two images appear on separate runs of the game.

1.1.4 Characterization

We initiate a more general study of extractable functions aimed at understanding extraction in of itself. In particular, we address questions such as: What makes a functions extractable? Is a function that is extractable in a weak sense extractable in a strong sense? Towards answering these questions, we give a set of three characterization theorems for interactive extraction with similar results for noninteractive extraction. These theorems relate notions of extraction with notions of obfuscation [BGI+01]. These theorems can be stated informally as follows.

1. *Any* function is either "weakly" extractable or "weakly" obfuscatable.

2. Any *"weakly-verifiable"* function is either "strongly" extractable or weakly obfuscatable. Moreover, any injective and strongly-extractable function is weakly verifiable.

3. Any *"weakly-verifiable"* function is either "very-strongly" extractable against a specific class, C, of adversaries or weakly obfuscatable . Moreover, if an efficiently computable and verifiable function is very-strongly extractable, then every adversary for this function is in C.

One of the main corollaries to the second theorem is that every POW function with auxiliary input is interactively extractable. This result supersedes the construction described in the previous section.

1.1.5 Applications

The final part of this thesis presents three applications of extractable function in very different settings.

1.1.5.1 Zero-knowledge

A zero-knowledge (ZK) protocol between a prover and a verifier [GMR85], allows the prover to convince the verifier of the validity of a statement without revealing anything else. One of the major efficiency criteria for such protocols is round complexity, i.e., number of messages sent in either direction. A main open problem in this area is construction of 3-round ZK protocols for any language in NP based on general computational assumptions [Bar01].

We use a variant of noninteractive extraction to construct 3-round ZK arguments for any language in NP. This construction uses the FLS technique [FLS99] on the extractable function and noninteractive witness-indistinguishable proofs [BOV03, GOS06]. The key point lies in the ability of a zero-knowledge simulator to recover information crucial for simulation by using an extractor on the private state of the verifier.

All previously known 3-round ZK constructions [HT98, HT99, BP04b, Lep02] require specific and nonstandard knowledge assumptions. On the other hand, our protocols are the first to be based on general (yet strong) computational assumptions without resorting to specific algebraic constructs.

1.1.5.2 Random Oracle Instantiation

The Random Oracle methodology [FS86, BR93] consists of designing a protocol in an idealized model (the random oracle model) and then moving this protocol to the standard model. The random oracle model allows each party oracle access to a random function. Whereas, the first step in this methodology is sound, the second step (called, instantiation) remains a heuristic for the most part, without proper justification for security in the standard model. In this context, we use extractable functions to replace random oracles, *while maintaining security*, in specific encryption schemes such as OAEP [BR94] and the encryption scheme of [BR93] as well as in a more general class of encryption schemes.

We emphasize that the contribution of this work is not in giving more efficient constructions than existing ones [Sah99, DDN00], but rather in making the Random Oracle methodology more rigorous for these schemes. For instance, our results yield the first full instantiation of OAEP. A different contribution is in designing new instantiation techniques that permit instantiating schemes that are provably unrealizable otherwise

[CGH98].

1.1.5.3 Obfuscating Multibit Point Functions

Obfuscation [BGI+01] refers to the ability of a code to compute a functionality without revealing anything about this functionality beyond the input/output behavior. Obfuscation remains a field dominated by the impossibility results of [BGI+01] with few positive constructions. Specifically, all previous constructions are for point functions and other related functions. A point function outputs 1 on a single input and 0 everywhere else.

In our final application, we exploit techniques used in constructing interactively-extractable functions to obfuscate multibit point functions and other related functions. A multibit point functions returns a long string on a single point and 0 everywhere else. This obfuscation can be applied to designing digital lockers, that is symmetric encryption with "weak" keys or passwords.

Previous obfuscation of multibit point functions either restrict the output to logarithmic length [Wee05] or the input distribution to uniform [FKSW05]. On the other hand, we give the first general obfuscation of multibit point functions.

1.2 Organization

This thesis is logically divided into two parts, theory and applications, with an additional chapter (Chapter 2) that gives common definitions and notations.

The theory is developed in Chapters 3, 4, and 5. Chapter 3 introduces, defines, and constructs noninteractive extraction. Chapter 4 does the same for interactive extraction. Characterization is presented in Chapter 5. Chapters 3 and 4 can be read independently. However, it is recommended that Chapter 5 is read after Chapters 3 and 4.

Chapters 6 and 7 presents the applications to zero-knowledge and random oracle instantiation, respectively. Each one of these chapters depends only on definitions of extraction in Chapters 3 and 4. Obfuscation of multibit point functions appears in Chapter 8. Chapter 8 is self-contained but uses some definitions from Chapter 2.

Chapter 2

Technical Preliminaries

We recall from the literature basic notation and common definitions that are used throughout this thesis.

2.1 Notations and Basic Definitions

A function, μ, is called **negligible** if it decreases faster than any inverse polynomial. Formally, for any polynomial p, there exists an integer N_p such that, for all $n \geq N_p$: $\mu(n) < \frac{1}{p(n)}$. We reserve μ to denote negligible functions.

If A is a set, then $a \leftarrow A$ means a is chosen uniformly at random from A. If D is a distribution, then $a \leftarrow D$ means a is sampled according to D. We denote by U_n the uniform distribution on $\{0,1\}^n$. A distribution is called **well-spread** if it has superlogarithmic min-entropy, i.e., $max_k Pr[X_n = k]$ is a negligible function in n.

A probabilistic function family is a set of efficient probabilistic functions having common input and output domains. Formally, $\mathbf{H}^n = \{H_k\}_{k \in K_n}$ is a function family with key space K_n and randomness domain R_n if, for all $k \in K_n, H_k : I_n \times R_n \rightarrow O_n$. A probabilistic function family has **public randomness** if for all k, $H_k(x,r) = (r, H'_k(x,r))$ for some deterministic function H'_k. A family ensemble is a collection of function families, i.e., $\mathbb{H} = \{\mathbf{H}^n\}_{n \in \mathbb{N}}$.

Let PPT denote any probabilistic polynomial-time Turing machine and nonuniform PPT any probabilistic polynomial-size circuit family. A PPT (respectively nonuniform PPT) A with oracle access to O is denoted by A^O.

2.2 One-way Functions and Uninvertible Functions

A one-way function, F, with respect to a well-spread distribution, $\mathbb{X}$, is an efficiently-computable function that is hard to invert approximately on $\mathbb{X}$ [DH76]. In other words, it is hard to find *any preimage* for a given value in the range. Formally,

Definition 2.2.1 (one-way function, [DH76]). *A function, F, is called one-way with respect to a well-spread distribution, $\mathbb{X}$, if for any PPT, A:*

$$Pr[x \leftarrow X_n,\ F(A(F(x))) = F(x)] < \mu(n).$$

F is called one-way if it is one-way with respect to the uniform distribution.

Note that for a deterministic function, f, one can decide whether y is an image of x under f by computing $f(x)$ and comparing y to $f(x)$. However, this method does not apply for probabilistic functions. Consequently, probabilistic functions are usually coupled with efficient verifiers to decide this relationship. The next definition of probabilistic one-way functions use such verifiers.

Definition 2.2.2 (probabilistic one-way function). *A probabilistic function, F (with randomness domain R_n), with a corresponding deterministic verifier, V_F, is called one-way with respect to a well-spread distribution, $\mathbb{X}$, if for any PPT, A:*

$$Pr[x \leftarrow X_n,\ r \leftarrow R_n,\ V_F(A(F(x,r)), F(x,r)) = 1] < \mu(n).$$

F is called one-way if it is one-way with respect to the uniform distribution.

An uninvertible function, F, with respect to a well-spread distribution, $\mathbb{X}$, is an efficiently computable function that is hard to invert exactly on $\mathbb{X}$. That is, it is hard to find the *same preimage* used in computing an image. Formally, for any PPT, A, $Pr[x \leftarrow X_n,\ A(F(x)) = x] < \mu(n)$. If F is uninvertible with respect to *any* well-spread distribution, then it is called uninvertible.

Moreover, a function in t inputs is called uninvertible if its output does not reveal any of its input. Formally, F is called uninvertible with respect to a vector of well-spread distributions, $\mathbb{X} = \{X^1, ..., X^t\}$, if for every PPT A:

$$Pr[(x_1, ..., x_t) \leftarrow (X_n^1, ..., X_n^t),\ y \leftarrow F(x_1, ..., x_t), x' \leftarrow A(y) : \exists i, x' = x_i] \leq \mu(n).$$

F is called uninvertible if it is uninvertible with respect to any vector of well-spread distributions (with the same arity).

Note that uninvertible functions differ from one-way functions in that it is hard to retrieve the *exact* input used to compute an image but not necessarily a point in the pre-image set, e.g., $f(x) = 0$ is uninvertible but not one-way.

2.3 Assumptions

2.3.1 The Discrete-Logarithm (DL) Assumptions

Assumption 2.3.1 (DL Assumption). *Let $\mathbb{PQG}$ denote the distribution on (p, q, g), where p and q are uniform primes such that $p = 2q+1$ and $|p| = n$, and g is a generator for the quadratic residue group (modulo p). Then, for any nonuniform PPT, A:*

$$Pr[(p, q, g) \leftarrow PQG_n,\ a \leftarrow Z_q,\ a' \leftarrow A(p, q, g, g^a) : a = a'] \leq \mu(n).$$

A stronger version of this assumption requires the last inequality to hold for *any* p,q,g. This assumption is used in [HT98] as well as this thesis for constructing 3-round zero-knowledge protocols.

Assumption 2.3.2 (Strong DL Assumption, [HT98]). *For every n, there is a tuple (p, q, g), where p and q are primes such that $p = 2q+1$ and $|p| = n$, and g is a generator for the quadratic residue group (modulo p), such that for any nonuniform PPT, A:*

$$Pr[a \leftarrow Z_q,\ a' \leftarrow A(p, q, g, g^a) : a = a'] \leq \mu(n).$$

2.3.2 The Decisional Diffie-Hellman (DDH) Assumption

Assumption 2.3.3 (DDH). *Let $\mathbb{PQG}$ denote the distribution on (p, q, g), where p and q are uniform primes such that $p = 2q + 1$ and $|p| = n$, and g is a generator for the quadratic residue group modulo p. Then for any PPT, A:*

$$|Pr[(p, q, g) \leftarrow PQG_n,\ x, y \leftarrow Z_q^*, Z_q^*,\ b \leftarrow A(p, q, g, g^x, g^y, g^{xy}) : b = 1]-$$

$$(p, q, g) \leftarrow PQG_n,\ x, y, z \leftarrow Z_q^*, Z_q^*, Z_q^*,\ b \leftarrow A(p, q, g, g^x, g^y, g^z) : b = 1]| \leq \mu(n).$$

2.4 Pseudorandom Generators

A pseudorandom generators stretches a uniform seed into a longer string that is computationally indistinguishable from uniform. Formally,

Definition 2.4.1 (Pseudorandom Generator, [BM84]). *A function, G, is a pseudorandom generator if:*

1. *G is efficiently computable.*
2. *$|G(x)| > |x|$ for all x.*
3. *For any nonuniform PPT, A:*

$$|Pr[x \leftarrow U_n,\ b \leftarrow A(G(x)) : b = 1] - Pr[b \leftarrow A(U_{|G(x)|}) : b = 1]| \leq \mu(n).$$

Definition 2.4.2 (Family of Pseudorandom Generators). *A family of functions, $\mathbb{G} = \{\{G_k\}_{k \in K_n}\}_{n \in \mathbb{N}}$, is a family of pseudorandom generator if:*

1. *G_k is efficiently computable for any $k \in K_n$, $n \in \mathbb{N}$.*
2. *$|G_k(x)| > |x|$ for all k and x.*
3. *For any PPT A:*

$$|Pr[k \leftarrow K_n,\ x \leftarrow U_n,\ b \leftarrow A(G_k(x)) : b = 1] -$$

$$Pr[k \leftarrow K_n,\ b \leftarrow A(U_{|G_k(x)|}) : b = 1]| \leq \mu(n).$$

2.5 Perfectly One-way Probabilistic Functions

A perfectly one-way (POW) function is a probabilistic function that hides all information about its input. Due to its probabilistic nature, such a function is coupled with an efficient verification scheme that determines whether a given string is a valid image of some given input. Moreover, we require that it satisfies collision resistance, i.e., it is hard to find two distinct input strings and an output string that is a valid image of each one of them. Efficient verification and collision resistance are formalized as follows.

Definition 2.5.1 (Efficient Verification, [Can97]). *A family ensemble, $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$, where for some polynomial $l(n)$, for any $n \in \mathbb{N}$, and any $k \in K_n$, $H_k : \{0,1\}^n \times R_n \to \{0,1\}^{l(n)}$, satisfies efficient verification if there exists a deterministic polynomial time algorithm, $V_{\mathbb{H}}$[1], such that:*

$$\forall k \in K_n,\ x \in \{0,1\}^n,\ r \in R_n,\ \ V_{\mathbb{H}}(x, H_k(x,r)) = 1.$$

A family ensemble that satisfies efficient verification is called *verifiable* for short.

Definition 2.5.2 (Collision Resistance, [Can97]). *A verifiable family ensemble, $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$, where for some polynomial $l(n)$, for any $n \in \mathbb{N}$, and any $k \in K_n$, $H_k : \{0,1\}^n \times R_n \to \{0,1\}^{l(n)}$, satisfies collision resistance if for any nonuniform PPT A:*

$$Pr[k \leftarrow K_n, (x_1, x_2, y) \leftarrow A(k) :\ x_1 \neq x_2 \text{ and } V_{\mathbb{H}}(x_1, y) = V_{\mathbb{H}}(x_2, y) = 1] \leq \mu(n).$$

2.5.1 Perfect One-wayness.

Hiding all information about the input has its roots in semantic security of probabilistic encryption [GM84] which requires that every function that can be computed given the ciphertext can also be computed without it. However, the notion of secrecy in this setting is slightly weaker than semantic security because an image can be used to verify the correctness of a guess. This notion is captured by a simulation-based definition. Informally, every predicate computable given an image can also be computed by a simulator with access to an oracle, I_x, where I_x accepts a query if and only if it matches x. The formal definition appears in [Can97].

There is another notion of perfect one-wayness that is easier to work with in the context of this thesis. This notion requires indistinguishability between images of the same input and some distribution.

Both notions can be formulated against unbounded adversaries (information-theoretic setting) or against PPT adversaries (computational setting). In the information-theoretic setting, these two notion are equivalent [DS05]. In the computational setting, the equivalence hold for a simpler notion of indistinguishability [Can97]. In the rest of the thesis we use the second notion.

[1] Even though, we don't explicitly include k in the input for V, we implicitly assume that it receives it.

We also consider the presence of auxiliary information in the computational setting. This auxiliary information is represented as an uninvertible function of the input.

Statistical Perfect One-wayness.

Statistical information hiding is captured by requiring statistical closeness between images of the same input and those of uniform inputs. Formally,

Definition 2.5.3 (Statistical t-Indistinguishability, [DS05]). *A verifiable family ensemble $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$, where $H_k : \{0,1\}^n \times R_n \to \{0,1\}^{l(n)}$ for some polynomial l, is called **statistically t-indistinguishable** if for any well-spread distribution $\mathbb{X} = \{X_n\}_{n\in\mathbb{N}}$ and any $k \in K_n$,*

$$\Delta(\underbrace{H_k(X_n, R_n^1), ..., H_k(X_n, R_n^{t(n)})}_{t(n)}, \underbrace{H_k(U_n^1, R_n^1), ..., H_k(U_n^{t(n)}, R_n^{t(n)})}_{t(n)}) \leq \mu(n),$$

where each distribution R_n^i (respectively, U_n^i) is the same as R_n (respectively, U_n).

Moreover, if $\mathbb{H}$ is statistically t-indistinguishable for any polynomial t then it is called statistically indistinguishable.

A special case of statistical indistinguishability is statistical pseudorandomness, where the images are indistinguishable from uniform. This notion is similar to the notion of extractors [DS05]. An extractor is a randomized function that takes inputs of high entropy and outputs strings statistically close to uniform. In other words, it "extracts" the randomness from the input to compress it into an almost uniform string.

Definition 2.5.4 (Statistical t-Pseudorandomness, [DS05]). *A verifiable family ensemble $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$, where $H_k : \{0,1\}^n \times R_n \to \{0,1\}^{l(n)}$ for some polynomial l, is called **statistically t-pseudorandom** if for any well-spread distribution $\mathbb{X} = \{X_n\}_{n\in\mathbb{N}}$ and any $k \in K_n$, $\Delta(\underbrace{H_k(X_n, R_n^1), ..., H_k(X_n, R_n^{t(n)})}_{t(n)}, \underbrace{U_n^1, ..., U_n^{t(n)}}_{t(n)}) \leq \mu(n)$, where each distribution R_n^i (respectively, U_n^i) is the same as R_n (respectively, U_n).*

Moreover, if $\mathbb{H}$ is statistically t-pseudorandom for any polynomial t then it is called statistically pseudorandom.

Computational Perfect One-wayness. Computational perfect one-wayness differs from statistical perfect one-wayness in two main ways. The first and obvious difference is that indistinguishability holds for *polynomially-bounded adversaries* only. Second,

computational perfect one-wayness may take the presence of auxiliary information into account. In this context, we restrict the notion of auxiliary information to uninvertible functions about the input. This restriction is necessary because otherwise auxiliary information reveals the input violating indistinguishability.

Instead of explicitly writing two definitions, one with auxiliary information and another without it, we present here one definition only. To take both cases into account, we use the convention that auxiliary information is surrounded by boxes. So, by removing $\boxed{\text{the words in boxes}}$ from Definition 2.5.5, we get the first definition while keeping the boxes yields the second one.

Definition 2.5.5 (t-Indistinguishability, [CMR98]). *Let F be any (possibly probabilistic) uninvertible function. A verifiable family ensemble $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$, where $H_k : \{0,1\}^n \times R_n \to \{0,1\}^{l(n)}$ for some polynomial l, is called* ***t-indistinguishable*** $\boxed{\textit{with auxiliary input } F}$ *if for any well-spread distribution, $\mathbb{X} = \{X_n\}_{n\in\mathbb{N}}$, any $k \in K_n$ and any PPT A:*

$$|Pr[x \leftarrow X_n,\ \boxed{z \leftarrow F(x)},\ (r_1,...,r_t) \leftarrow (R_n,...,R_n) :$$

$$A(k, \boxed{z}, H_k(x,r_1),...,H_k(x,r_t)) = 1] \quad -$$

$$Pr[x \leftarrow X_n,\ (u_1,...,u_t) \leftarrow (U_n,...,U_n),\ \boxed{z \leftarrow F(x)},\ (r_1,...,r_t) \leftarrow (R_n,...,R_n) :$$

$$A(k, \boxed{z}, H_k(u_1,r_1),...,H_k(u_t,r_t)) = 1]| \leq \mu(n).$$

$\boxed{\textit{If } \mathbb{H} \textit{ is t-indistinguishable with any auxiliary input } F, \textit{ then it is called}}$ $\boxed{\textit{t-indistinguishable with auxiliary input}}$. *Moreover, if it is t-indistinguishable* $\boxed{\textit{with auxiliary input}}$ *for any polynomial t, then it is called indistinguishable* $\boxed{\textit{with auxiliary input}}$.

A special case of computational indistinguishability is computational pseudorandomness, i.e., images of the same input are indistinguishable from uniform. Formally,

Definition 2.5.6 (t-Pseudorandomness, [CMR98]). *Let F be any (possibly probabilistic) uninvertible function. A verifiable family ensemble $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$, where $H_k : \{0,1\}^n \times R_n \to \{0,1\}^{l(n)}$ for some polynomial l, is called* ***t-pseudorandom*** $\boxed{\textit{with auxiliary input } F}$ *if for any well-spread distribution, $\mathbb{X} = \{X_n\}_{n\in\mathbb{N}}$, any $k \in K_n$,*

any PPT A:

$$|Pr[x \leftarrow X_n, \boxed{z \leftarrow F(x)}, (r_1, ..., r_t) \leftarrow (R_n, ..., R_n) :$$

$$A(k, \boxed{z}, H_k(x, r_1), ..., H_k(x, r_t)) = 1] \quad -$$

$$Pr[x \leftarrow X_n, \boxed{z \leftarrow F(x)} : A(k, \boxed{z}, U_{lt}) = 1]| \leq \mu(n).$$

If $\mathbb{H}$ is t-pseudorandom with any auxiliary input F, then it is called t-pseudorandom $\boxed{\textit{with auxiliary input}}$. *Moreover, if it is t-pseudorandom* $\boxed{\textit{with auxiliary input}}$ *for any polynomial t, then it is called pseudorandom* $\boxed{\textit{with auxiliary input}}$.

2.6 Obfuscation

We adopt the definition of obfuscation used in [Can97, Wee05] because obfuscation of point functions is known for this notion only (if the distribution on this class of functions is not restricted). This definition is weaker than the one in [BGI+01] because the size of the simulator is allowed to depend on the quality of the simulation. We note that the impossibility results of [BGI+01] applies for this notion also. The formal definition follows.

Definition 2.6.1 (Obfuscation, [BGI+01, Can97, Wee05]). *Let $\mathbb{F}$ be any family of functions. A PPT, O, is called an* ***obfuscator*** *of $\mathbb{F}$, if:*

1. ***Approximate Functionality:*** *For any $F \in \mathbb{F}$: $Pr[\exists x, O(F)(x) \neq F(x)]$ is negligible. Here, the probability is taken over the coin tosses of O.*

2. ***Polynomial Slowdown:*** *There is a polynomial p such that, for any $F \in \mathbb{F}$, $O(F)$ runs in time at most $p(T_F)$, where T_F is the worst-case running time of F.*

3. ***Virtual Black-box Property:*** *For any nonuniform PPT A and any polynomial p, there exists a nonuniform PPT S such that for any $F \in \mathbb{F}$ and sufficiently large n:*

$$|Pr[b \leftarrow A(O(F)) : b = 1] - Pr[b \leftarrow S^F(1^{|F|}) : b = 1]| \leq \frac{1}{p(n)}.$$

2.7 Encryption Schemes

We recall the definitions of indistinguishability under chosen plaintext (IND-CPA) (which is equivalent to semantic security [GM84]) and chosen ciphertext attack (IND-CCA2). These definitions remain the same in the Random Oracle model except that every PPT has oracle access to a random function, denoted by O.

Definition 2.7.1 (IND-CPA, [GM84]). *A public key encryption scheme, (G, E, D), is called IND-CPA if for any PPT pair (A_1, A_2):*

$$|Pr[(pk, sk) \leftarrow G(1^n),\ (m_0, m_1, s) \leftarrow A_1(pk),\ c \leftarrow E(m_0, pk),\ b \leftarrow A_2(s, c) : b = 1] \ -$$

$$Pr[(pk, sk) \leftarrow G(1^n),\ (m_0, m_1, s) \leftarrow A_1(pk),\ c \leftarrow E(m_1, pk),\ b \leftarrow A_2(s, c) : b = 1]| \leq \mu(n).$$

Definition 2.7.2 (IND-CCA2, [NY90]). *A public key encryption scheme, (G, E, D), is called IND-CCA2 if for any PPT pair $(A_1^{D(.,sk)}, A_2^{D(.,sk)})$:*

$$|Pr[(pk, sk) \leftarrow G(1^n),\ (m_0, m_1, s) \leftarrow A_1^{D(.,sk)}(pk),\ c \leftarrow E(m_0, pk),\ b \leftarrow A_2^{D(.,sk)}(s, c) :$$

$$b = 1] \ -$$

$$Pr[(pk, sk) \leftarrow G(1^n),\ (m_0, m_1, s) \leftarrow A_1^{D(.,sk)}(pk),\ c \leftarrow E(m_1, pk),\ b \leftarrow A_2^{D(.,sk)}(s, c) :$$

$$b = 1]| \leq \mu(n),$$

where it is assumed that $A_2^{D(.,sk)}$ does not query $D(., sk)$ on c.

2.8 Zero-knowledge Arguments

A zero-knowledge (ZK) argument system [GMR85] consists of a polynomial-time prover, P interacting with a polynomial-time verifier, V. The prover is given a theorem, x, and a witness for its correctness, w while V receives only x. The prover has to convince V of the validity of x *without revealing anything beyond the validity of* x (see also introduction of Chapter 6). Formally,

Definition 2.8.1 (Zero-knowledge Argument System, [GMR85]). *Let L be an NP language (with relation R_L). Then $\mathcal{P} = (P, V)$ is called a* ***zero-knowledge argument***

system *(ZK for short) for L if both P and V are PPT, and the following three conditions hold:*

1. ***Completeness.*** *For every $(x, w) \in R_L$,*

$$Pr[b \leftarrow < P(x, w), V(x) > : b = 1] \geq 1 - \mu(n).$$

2. ***Soundness.*** *For every PPT, $\hat{P}$, and any $x \notin L$:*

$$Pr[b \leftarrow < \hat{P}(x), V(x) > : b = 1] \leq \mu(n).$$

3. ***Zero-knowledge.*** *For any PPT, $\hat{V}$, there exists a PPT machine, S, such that for any PPT distinguisher, D, any $(x, w) \in R_L$, and any distribution, $\mathbb{Z} = \{Z_n\}_{n \in \mathbb{N}}$:*

$$|Pr[z \leftarrow Z_n,\ b \leftarrow D(< P(x, w), \hat{V}(x, z) >) : b = 1] -$$

$$Pr[z \leftarrow Z_n,\ b \leftarrow D(S(x, z)) : b = 1]| \leq \mu(n).$$

2.9 Zero-knowledge Proofs of Knowledge

Zero-knowledge proofs of knowledge [GMR85, BG92] require in addition to Definition 2.8.1 that the prover knows a witness. In other words, if the verifier is convinced, then there is an extractor that recovers a witness from the prover. Formally,

Definition 2.9.1 (Zero-knowledge proofs of knowledge, [GMR85, BG92]). *Let L be an NP language (with relation R_L). Then $\mathcal{P} = (P, V)$ is called a* ***zero-knowledge proof of knowledge*** *for L if $\mathcal{P}$ is a zero-knowledge argument for L and the following holds:*

Proof of knowledge: *For any PPT, $\hat{P}$ (with random coins $r_{\hat{P}}$ and randomness domain $R_{\hat{P}}$), there exists an extractor $\mathcal{K}_{\hat{P}}$ such that for any x:*

$$Pr[r_{\hat{P}} \leftarrow R_{\hat{P}},\ b \leftarrow < \hat{P}(x, r_{\hat{P}}), V(x) >,\ w \leftarrow \mathcal{K}_{\hat{P}}(x, r_{\hat{P}}) : b = 1 \text{ and } (x, w) \notin R_L] \leq \mu(n).$$

2.10 Non-interactive Zero-knowledge Arguments

In a noninteractive argument system, the prover has to convince the verifier of the validity of a statement by sending a single message only, called the proof. Such protocols need a setup assumption, namely, a randomly-generated, public string called the **Common Reference String** (CRS for short). Formally,

Definition 2.10.1 (Noninteractive Zero-knowledge Argument System, [BFM88]). *Let L be an NP language (with relation R_L). Then $\mathcal{P} = (P, V)$ is called a* ***noninteractive zero-knowledge argument system*** *(NIZK for short) for L if both P and V are PPT, and the following three conditions hold:*

1. ***Completeness.*** *For every $(x, w) \in R_L$,*

$$Pr[\sigma \leftarrow U_n,\ \pi \leftarrow P(x, w, \sigma),\ b \leftarrow V(x, \pi, \sigma) : b = 1] \geq 1 - \mu(n).$$

2. ***Soundness.*** *For every PPT, $\hat{P}$:*

$$Pr[\sigma \leftarrow U_n,\ (x, \pi) \leftarrow \hat{P}(\sigma),\ b \leftarrow V(x, \pi, \sigma) : b = 1 \text{ and } x \notin L] \leq \mu(n).$$

3. ***Zero-knowledge.*** *There exists a PPT pair, $S = (S_1, S_2)$, such that for any polynomial, t, any PPT, A, any $(x_1, w_1), ..., (x_{t(n)}, w_{t(n)}) \in R_L$ (that may depend on σ), and any distribution, $\mathbb{Z} = \{Z_n\}_{n \in \mathbb{N}}$:*

$$|Pr[\sigma \leftarrow U_n,\ z \leftarrow Z_n,$$

$$b \leftarrow A(z, (x_1, P(x_1, w_1, \sigma)), ..., (x_{t(n)}, P(x_{t(n)}, w_{t(n)}, \sigma)), \sigma) : b = 1] -$$

$$Pr[(\sigma, aux(\sigma)) \leftarrow S_1(1^n),\ z \leftarrow Z_n,$$

$$b \leftarrow A(z, (x_1, S(x_1, \sigma, aux(\sigma))), ..., (x_{t(n)}, S(x_{t(n)}, \sigma, aux(\sigma))), \sigma) : b = 1]| \leq \mu(n).$$

2.11 Non-interactive Witness Indistinguishable Arguments

Although NIZK arguments for all languages in NP require a CRS [BFM88], if we relax the zero knowledge property to witness indistinguishability, we can realize it without a

setup assumption [BOV03, GOS06]. Witness indistinguishability means that it is hard to tell which witness is used in generating a proof. Formally,

Definition 2.11.1 (Noninteractive Witness Indistinguishable Argument System, [FS90, BOV03, GOS06]). *Let L be an NP language (with relation R_L). Then $\mathcal{P} = (P, V)$ is called a* ***noninteractive witness-indistinguishable (WI) argument system*** *for L if both P and V are PPT, and the following three conditions hold:*

1. ***Completeness.*** *For every $(x, w) \in R_L$,*

$$Pr[\pi \leftarrow P(x, w),\ b \leftarrow V(x, \pi) : b = 1] \geq 1 - \mu(n).$$

2. ***Soundness.*** *For every PPT, $\hat{P}$:*

$$Pr[(x, \pi) \leftarrow \hat{P}(1^n),\ b \leftarrow V(x, \pi) : b = 1 \text{ and } x \notin L] \leq \mu(n).$$

3. ***Witness Indistinguishability.*** *For any PPT, A, any polynomial t, and any $(x_1, w_1^1), (x_1, w_1^2), ..., (x_{t(n)}, w_{t(n)}^1),\ (x_{t(n)}, w_{t(n)}^2) \in R_L$, and any distribution $\mathbb{Z} = \{Z_n\}_{n \in \mathbb{N}}$:*

$$|Pr[z \leftarrow Z_n,\ \pi_1, ..., \pi_{t(n)} \leftarrow P(x_1, w_1^1), ..., P(x_{t(n)}, w_{t(n)}^1),$$

$$b \leftarrow A(z, (x_1, w_1^1, w_1^2, \pi_1), ..., (x_{t(n)}, w_{t(n)}^1, w_{t(n)}^2, \pi_{t(n)})) : b = 1] -$$

$$Pr[z \leftarrow Z_n,\ \pi_1, ..., \pi_{t(n)} \leftarrow P(x_1, w_1^2), ..., P(x_{t(n)}, w_{t(n)}^2),$$

$$b \leftarrow A(z, (x_1, w_1^1, w_1^2, \pi_1), ..., (x_{t(n)}, w_{t(n)}^1, w_{t(n)}^2, \pi_{t(n)})) : b = 1]| \leq \mu(n).$$

2.12 Σ-Protocols

A Σ-protocol is a 3-round *honest-verifier* Arthur-Merlin proof of knowledge. The prover starts the protocol by sending a message a, the verifier responds with a random string e, and the prover sends z in the last round. A Σ-protocol has a special soundness properties that allows for witness extraction. Specifically, for any x and any two accepting conversations (a, e, z) and (a, e', z') for $e \neq e'$, it is easy to compute a witness for x. The

honest-verifier zero-knowledge property says that whatever an honest verifier computes from a conversation can be computed without it. Formally:

Definition 2.12.1 (Σ-Protocol, [Blu86]). *A 3-round Arthur-Merlin game, $\mathcal{P} = (P, V)$, is called a Σ-protocol for a language L (with NP relation R_L) if:*

1. ***Completeness.*** *For any $(x, w) \in R_L$, $Pr[b \leftarrow< P(x, w), V(x) >: b = 1] = 1$, where $< V(x), P(x) >$ denotes the output of V at the end of the protocol.*

2. ***Special Soundness.*** *There is a PPT, $\mathcal{K}$, such that for any x and any two accepting conversations, (a, e, z) and (a, e', z') for $e \neq e'$, $\mathcal{K}(x, (a, e, z), (a, e', z')) = w$ where $(x, w) \in R_L$.*

3. ***Honest-verifier Zero-Knowledge.*** *Let $view_V(x, e)$ denote the view of V on input x and public randomness e (the second message). Then, there exists a PPT, S, such that for any $x \in L$, $view_V(x, e)$ and $S(x, e)$ have the same distribution.*

We remark that every NP language has a Σ-protocol if one-way functions exist [Blu86]. Moreover, the prover is efficient if it is supplied with a witness.

Chapter 3

Extractable Functions

Summary: We introduce and formalize a notion of computational knowledge, called **extractable functions**, and give several constructions.

Informally, an extractable function guarantees that any machine that produces a point in the range, knows a corresponding preimage. This knowledge is captured by the existence of an efficient machine, called the extractor, that recovers the preimage in question. We formalize this notion in several models. We consider extraction for a single function and extraction for a family of functions. In the latter case, a function is chosen randomly from the family and given to the adversary. We also consider models with and without auxiliary information. There are three cases:

1. There is no auxiliary information.
2. There is independent auxiliary information. Here the dependency is on the function itself and is relevant when extraction is for a family of functions.
3. There is dependent auxiliary information.

After formalizing this notion, we present several constructions. The constructions satisfy two properties. The first one is a knowledge property and is usually extraction for a family of functions in the presence of independent auxiliary information. The second property is a computational-hardness property and can be one-wayness, pseudorandomness, or perfect one-wayness.

This chapter is based on the paper [CD08a], which is a joint work with Ran Canetti. Note that [CD08a] contains some additional results that do not appear in this chapter.

All but the last construction, utilize, in addition to a hardness assumption, a knowledge assumption such as the Knowledge of Exponent (KE) assumption (see Assumption 3.3.1). The final construction is based on a variant of noninteractive zero-knowledge (NIZK) arguments of knowledge. In fact, we show the equivalence between these two primitives.

3.1 Introduction

An extractable function is one for which any machine that computes a point in the range knows a corresponding preimage. In other words, there is a family of functions and the adversary gets a description of a specific function from the family, and tries to output a point in the range. This function family is considered noninteractively extractable if whenever the adversary generates a value in the range, it knows a preimage. That is, for every such adversary there is a corresponding extractor that computes a preimage from the private input of the adversary. One extreme example of extractable functions is the identity function where the output itself reveals the input. Obviously, such functions are of lesser interest to cryptographic applications than functions with computational hardness properties. On another extreme, if the function is a one-way permutation, then it is easy to output a valid image without knowing a preimage; specifically, output a random string in the range. In this thesis, we concentrate on functions that enjoy both properties, namely, extractability and computational hardness.

From a different angle, extractability can be interpreted as: the only way to produce a value in the range of a function is by taking a point in the input domain and then applying the algorithm that computes this function to the input. In other words, extractability reduces adversarial strategies to honest-but-curious strategies (strategies that follow a prescribed protocol but compute something extra on the side).

On efficient verification. Unlike proofs of knowledge [GMR85, BG92], this notion of extraction does not *require* efficient verification. In other words, the range of the function is not necessarily efficiently verifiable. Therefore, it may not be possible to decide if the adversary generates a point in the range (and consequently, knows a preimage). However, this notion guarantees the implication: If the adversary generates an image, it knows a preimage. We mention that Construction 3.3.2 has a range that is efficiently verifiable

in the presence of some auxiliary information about the function itself.

On the relation between extractable functions and knowledge assumptions. We view *extractable functions as an abstraction away from specific knowledge assumptions such as the knowledge of exponent (KE) assumption [Dam92, HT98] and the proof of knowledge (POK) assumption [Lep02], much like a one-way function is an abstraction of specific one-way assumptions, such as the discrete logarithm (DL) assumption.* In other words, the DL assumption gives us a one-way function but it may even give us more, e.g., a one-way permutation in certain groups or with certain algebraic properties. However, we abstract away from these particularities and identify the essential property needed. Likewise, we use extractable functions as a step towards capturing the abstract knowledge assumption - it provides a relatively simple primitive that is defined only in terms of its general computational properties, that is useful in a number of places, and that can be realized by a number of different assumptions.

On the relation between extractable functions and NIZK. Superficially, extractable functions resemble noninteractive zero-knowledge (NIZK) proofs of knowledge [SP92, SCO+01] in that an image can be viewed as a proof of preimage knowledge. However, proofs of knowledge are weaker. This is so because NIZK proofs of knowledge require a universal blackbox extractor to recover a witness *with the help of auxiliary information about the common reference string (CRS).* On the other hand, extractable functions require a nonblackbox extractor for every adversary. However, this extractor has to recover a preimage from the view of the adversary *without any extra information that is not given to the adversary.* The latter formulation may better capture our intuition about knowledge because it clearly demonstrates that an adversary knows a preimage by recovering it from its view alone. In fact, we show in Section 3.4 that a stronger notion of NIZK proofs of knowledge, where extraction occurs in a nonblackbox way and without auxiliary information about the CRS, is equivalent to an extractable function that satisfies some form of perfect one-wayness.

3.1.1 Our Work

This chapter is devoted solely to formalizing and constructing noninteractively-extractable functions.

3.1.1.1 Formulating Extraction

The general format of a definition of extraction is as follows: for any efficient adversary, A, there is an efficient extractor, $\mathcal{K}_A$, that depends on A and has access to the private input of A, including its random coins. Moreover, $\mathcal{K}_A$ has negligible failure error; that is the probability that the output of A is valid but the output of $\mathcal{K}_A$ is not a valid preimage is negligibly close to 0.

There are *five* variants of this general definition depending on two major criteria. First, extraction can be required for any function in the family or for a uniformly chosen function. In the latter case, the probability of extraction is taken over the choice of the function. The constructions that we give in this thesis satisfy the latter notion. For completeness, we present the former notion as well.

Second, extraction can be formalized with or without auxiliary information. We consider extraction in the presence of auxiliary information as this is a more useful and meaningful notion. Auxiliary information can be either dependent or independent [GK05] (here, the dependence is on the specific function under study). We remark that dependent auxiliary information is inseparable from independent auxiliary information when extraction is required for a single function, f. This is so because it is not possible to prevent an adversary with access to auxiliary information from receiving dependent auxiliary information, e.g., $f(x)$. Moreover, the notion of a single extractable function with auxiliary information is not realizable for one-way functions. Specifically, by the one-wayness assumption, there is no extractor for the adversary that receives $f(x)$, for a uniform x, and simply outputs it. Consequently, the notion of extraction with auxiliary information is meaningful only for a function family. Indeed, the KE assumption is formulated in terms of function families.

3.1.1.2 Constructions

We build extractable functions from four different sources of knowledge, specifically, the KE assumption (Assumption 3.3.1), the POK assumption (Assumption 3.3.4), the Diffie-Hellman proof of knowledge (DH-KEA) assumption (see [PX09] and Assumption 3.3.3), and NIZK proofs of knowledge. We also combine knowledge properties with hardness assumptions to yield extractable functions with computational properties such as one-wayness, pseudorandomness, and perfect one-wayness. Refer to Table 3.1 for a list of

	DL	DDH	strong DDH
KE	Extractable OW	Extractable PRG	Extractable POW
POK	Extractable OW	-	Extractable POW (1-indist.)
DH-KEA	Extractable OW	Extractable PRG	Extractable POW

Table 3.1: Constructions based on the KE Assumption. KE= Knowledge of Exponent, POK= Proof of Knowledge, DH-KEA=Diffie-Hellman Knowledge of Exponent, DL=Discrete Log, DDH= Decisional Diffie-Hellman, OW= One-way, PRG=Pseudorandom Generator, POW=Perfectly One-way.

the results and needed assumptions.

From the KE assumption. The KE assumption can be combined with:

- DL assumption to give an extractable one-way function.
- DDH assumption to give an extractable pseudorandom generator.
- strong version of DDH (see Definition 3.3.2) to give an extractable POW function.

Informally, the key construction utilizes the quadratic residue group modulo a safe prime.[1] Let g be a generator for this group and g^a be a uniform element for this group. Then, $H_{p,g,g^a}(x,r) = g^r, g^{ar}, g^{rx}, g^{arx}$. At a high level, the KE assumption allows us to recover r and rx (and consequently, x). Moreover, the strong DDH assumption gives us perfect one-wayness.

We mention that the DH-KEA assumption is stronger than the KE assumption [PX09], and consequently, it implies the results described here.

From the POK assumption. In a similar fashion, the POK assumption can be combined with the DL assumption to give an extractable one-way function and with the strong DDH assumption to give an extractable POW function. However, it does not seem to imply extractable pseudorandom generators because the output needed for extraction is easily distinguishable from uniform.

From NIZK proofs of knowledge. As we mentioned previously, if we strengthen NIZK proofs of knowledge, they become equivalent to extractable functions that satisfy some form of perfect one-wayness. In other words, the existence of either one of them implies the existence of the other (modulo the existence of other standard primitives such as encryption and witness-indistinguishable (WI) proofs [BOV03, GOS06]).

[1] A prime, p, is safe if it can be written as $2q+1$, where q is another prime.

In more detail, both the knowledge and secrecy (zero-knowledge) properties of NIZK are weaker than extractable POW functions. As previously discussed, proofs of knowledge require an extractor to work with access to some *private* auxiliary information about the CRS, which is not available to the prover. On the other hand, extractable functions require the views of the adversary and the extractor to be the same. Moreover, zero-knowledge in the noninteractive setting requires secrecy *over a randomly chosen CRS*, whereas POW functions require secrecy *for any function*. In Section 3.4, we show that if we strengthen NIZK proofs of knowledge so that the extractor has the same view as the prover, we can construct extractable functions that are perfectly one-way for a randomly-chosen function (weak POW functions). (The last property is inherited directly from the zero-knowledge property described above.) In the reverse direction, we construct such NIZK proofs of knowledge *for any language in NP* given any extractable weak POW function.

3.1.2 On the Strength of the Assumptions

With the exception of the construction from NIZK proofs of knowledge, all known constructions of nontrivial extractable functions require a knowledge assumption. These assumptions are usually described in the literature as strong. This is so because they are not efficiently falsifiable [Nao03]. That is, in order to refute such an assumption, one needs to find an adversary *and prove that no machine can recover a preimage of the output of this adversary*. The last task is regarded as inefficient because of the quantification over all machines. Contrast this with a standard one-way assumption. A one-way assumption on a candidate function seems easier to refute by exhibiting a specific adversarial strategy to invert the function. In spite of this classification of assumptions, Bellare and Palacio [BP04a] show how to refute a specific knowledge assumption.

In Chapter 4, we study a weaker notion of extraction (specifically, interactive extraction) that can be realized from efficiently-falsifiable assumptions.

3.1.3 Organization

We formalize extractable functions in Section 3.2, construct them in Section 3.3, and finally discuss the connection to NIZK proofs of knowledge in Section 3.4.

3.2 Definitions

As we mentioned in the introduction, an extractable function is one for which any machine that computes a point in the range, knows a corresponding preimage. As a starting point, we can formulate this notion by requiring any efficient machine that computes an image *without auxiliary input* to know a preimage. Although, this requirement seems reasonable, it is not sufficient for applications where auxiliary information is present. On the other hand, formulating this notion in the presence of auxiliary information is tricky. As a toy example, A can be a machine that receives an image as an input and copies it to its output. In another scenario, A may receive an image hidden in its auxiliary input in a subtle way but can be efficiently extracted from it. Yet, we do not think that this captures our intuition because A does not really compute the function, rather it decodes the image syntactically from its input. Thus, we need a meaningful way of telling apart "copying" an image from "computing" an image.

Following [GK05], we consider two types of auxiliary information. The first one, called **independent auxiliary information**, consists of auxiliary information independent of the particular function currently used. This prevents hiding images in this type of input. The second type, called **dependent auxiliary information**, may depend on the function. Here, the issue of distinguishing "copying" an image from "computing" an image arises due to possible encoding of images in this input. We solve this problem by restricting this dependency to include only images under the function being used. Even though this dependency is very restricted, it is sufficient for our applications.

Given these two types of inputs, we require that no adversary can come up with a *new* image without knowing a corresponding preimage. In other words, for every A, that computes a new image, there is a corresponding extractor, $\mathcal{K}_A$, that computes a preimage, given access to the private input of A. We emphasize that $\mathcal{K}_A$ has to compute the preimage from the view of A without any additional information.

For clarity, we first formalize this notion without auxiliary information, then in the presence of independent auxiliary information and finally we present the general case. Also, we give definitions of extraction for a fixed function and for a function chosen randomly from a family.

3.2.1 Preimage Knowledge without Auxiliary Information

The strongest definition of preimage knowledge requires extraction to work for any function from a family. Specifically,

Definition 3.2.1 (Noninteractive extraction without auxiliary information). *A verifiable family ensemble, $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$ (with verifier $V_{\mathbb{H}}$), is called **noninteractively extractable without auxiliary information** if for any PPT, A (with private random coins denoted by r_A), there exists a PPT, $\mathcal{K}_A$ such that for any $k \in K_n$:*

$$Pr[y = A(k, r_A),\ x \leftarrow \mathcal{K}_A(k, r_A) : V_{\mathbb{H}}(x, y) = 1 \text{ or } (\forall x', V_{\mathbb{H}}(x', y) \neq 1)] > 1 - \mu(n).$$

Here and in the rest of the thesis, all probabilistic experiments that contain r_A are taken over the random coins r_A unless specified otherwise.

We currently do not know of any nontrivial (e.g., one-way) family that satisfies this definition. However, we have constructions that satisfy a weaker notion where extraction holds if k is chosen randomly. Formally,

Definition 3.2.2 (Noninteractive extraction without auxiliary information). *A verifiable family ensemble, $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$, is called **noninteractively extractable without auxiliary information** if for any PPT, A (with private random coins denoted by r_A), there exists a PPT, $\mathcal{K}_A$ such that:*

$$Pr[k \leftarrow K_n,\ y = A(k, r_A),\ x \leftarrow \mathcal{K}_A(k, r_A) : V_{\mathbb{H}}(x, y) = 1 \text{ or } (\forall x', V_{\mathbb{H}}(x', y) \neq 1)]$$

$$> 1 - \mu(n).$$

3.2.2 Preimage Knowledge with Independent Auxiliary Information

Adding auxiliary information to Definition 3.2.1 results in a definition for dependent auxiliary information because k is fixed in advance. So, we present this case in the next section.

The corresponding version of Definition 3.2.2 adds only a distribution of independent auxiliary information, $\mathbb{Z} = \{Z_n\}_{n\in\mathbb{N}}$.

Definition 3.2.3 (Noninteractive extraction with independent auxiliary information). *A verifiable family ensemble, $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$, is called* ***noninteractively extractable with independent auxiliary information*** *if for any PPT, A (with private random coins denoted by r_A), there exists a PPT, $\mathcal{K}_A$, such that for any distribution, $\mathbb{Z} = \{Z_n\}_{n\in\mathbb{N}}$:*

$$Pr[k \leftarrow K_n,\ z \leftarrow Z_n,\ y = A(k, z, r_A),\ x \leftarrow \mathcal{K}_A(k, z, r_A) :$$

$$V_{\mathbb{H}}(x, y) = 1 \text{ or } (\forall x', V_{\mathbb{H}}(x', y) \neq 1)] > 1 - \mu(n).$$

3.2.3 Preimage Knowledge with Dependent Auxiliary Information

Recall from previous discussions that introducing auxiliary information into Definition 3.2.1 yields a definition with dependent auxiliary information, where the dependency is unrestricted. Such a definition is equivalent to Definition 3.2.3 except that the inequality holds *for any* k.

There are two possible ways to introduce dependent auxiliary information (in the restricted form described above) into Definition 3.2.3. One can allow this auxiliary information to be images of any input while the more restrictive way forces the images to correspond to inputs chosen from well-spread distributions. Even though the former is stronger, the latter is sufficient for our applications. We give both versions starting with the stronger.

Definition 3.2.4 (Noninteractive extraction with dependent auxiliary information). *A verifiable family ensemble, $\mathbb{H}$, is called* ***noninteractively t- extractable*** *(t-extractable, for short) with dependent auxiliary information if for any PPT, A (with private random input, r_A), there exists a PPT, $\mathcal{K}_A$, such that for any $x_1, ..., x_{t(n)}$, and any distribution, $\mathbb{Z} = \{Z_n\}_{n\in\mathbb{N}}$:*

$$Pr[k \leftarrow K_n,\ z \leftarrow Z_n,\ r_1, ..., r_{t(n)} \leftarrow R_n, ..., R_n,$$

$$y = A(k, z, H_k(x_1, r_1), ..., H_k(x_{t(n)}, r_{t(n)}), r_A),$$

$$x \leftarrow \mathcal{K}_A(k, z, H_k(x_1, r_1), ..., H_k(x_{t(n)}, r_{t(n)}), r_A) :$$

$$V_{\mathbb{H}}(x,y)=1 \text{ or } (\exists i,\ y=H_k(x_i,r_i)) \text{ or } (\forall x', V_{\mathbb{H}}(x',y)\neq 1)] > 1-\mu(n).$$

If $\mathbb{H}$ is t-extractable with dependent auxiliary information for every polynomial t, then it is called extractable with dependent auxiliary information.

Definition 3.2.5 (Noninteractive extraction with dependent auxiliary information). *A verifiable family ensemble, $\mathbb{H}$, is called* ***noninteractively t- extractable*** *(t-extractable, for short) with dependent auxiliary information if for any PPT, A (with private random input, r_A), there exists a PPT, $\mathcal{K}_A$, such that for any vector of polynomially-many well-spread distributions $\mathbb{X}=\{X^1,...,X^t\}$, any polynomial, t_2, and any uninvertible function, F:*

$$Pr[k \leftarrow K_n,\ (x_1,...,x_{t(n)}) \leftarrow X_n^1,...,X_n^{t(n)},\ z \leftarrow F(x_1,...,x_{t(n)}),$$

$$r_1^{x_1},...,r_{t_2(n)}^{x_1},...,r_{t_2(n)}^{x_{t(n)}} \leftarrow R_n,...,R_n,\ y=A(k,z,H_k(x_1,r_1^{x_1}),...,H_k(x_{t(n)},r_{t_2(n)}^{x_{t(n)}}),r_A),$$

$$x \leftarrow \mathcal{K}_A(k,z,H_k(x_1,r_1^{x_1}),...,H_k(x_{t(n)},r_{t_2(n)}^{x_{t(n)}}),r_A):$$

$$V_{\mathbb{H}}(x,y)=1 \text{ or } (\exists i,j,\ y=H_k(x_i,r_j^i)) \text{ or } (\forall x', V_{\mathbb{H}}(x',y)\neq 1)] > 1-\mu(n).$$

If $\mathbb{H}$ is t-extractable for every polynomial t, then it is called extractable.

Note that the independent auxiliary information, z, is allowed to depend on the preimages, $x_1,...,x_{t(n)}$.

Definition 3.2.5 has another formulation where A has access to an oracle that provides images of the same inputs but with new random coins (for the function) every time it is queried. That is, $O_{x_1,...,x_{t_1}}(i)=H_k(x_i,r)$ for new random coins r. Let *hist* denote the history of interaction between A and $O_{x_1,...,x_{t_1}}$. Then, the following definition is equivalent to Definition 3.2.5.

Definition 3.2.6 (Noninteractive extraction with dependent auxiliary information (alternative version)). *A verifiable family ensemble, $\mathbb{H}$, is called* ***noninteractively t- extractable*** *(t-extractable, for short) with dependent auxiliary information if for any PPT, A (with private random input, r_A), there exists a PPT, $\mathcal{K}_A$, such that for any vector of polynomially-many well-spread distributions $\mathbb{X}=\{X^1,...,X^{t_1}\}$, any*

polynomial, t_2, and any uninvertible function, F:

$$Pr[k \leftarrow K_n,\ (x_1, ..., x_{t_1(n)}) \leftarrow X_n^1, ..., X_n^{t_1(n)},\ z \leftarrow F(x_1, ..., x_{t_1(n)}),$$

$$y \leftarrow A^{O_{x_1,...,x_{t_1}}}(k, z, r_A),\ x \leftarrow \mathcal{K}_A^{O_{x_1,...,x_{t_1}}}(k, z, hist, r_A) :$$

$$V_{\mathbb{H}}(x, y) = 1 \text{ or } (y \in hist) \text{ or } (\forall x', V_{\mathbb{H}}(x', y) \neq 1)] > 1 - \mu(n).$$

If $\mathbb{H}$ is t-extractable for every polynomial t, then it is called extractable.

3.3 Constructions

We give constructions of extractable functions based on the KE assumption in Section 3.3.1, on the DH-KEA assumption in Section 3.3.2, and on the POK assumption in Section 3.3.3.

3.3.1 Constructions from the KE Assumption

We give three constructions from the KE assumption. All constructions satisfy extractability. However, each one satisfies a different computational hardness property. The first construction is one-way, the second is pseudorandom, while the last one is perfectly one-way. Table 3.1 lists these results with the needed assumptions.

Before we present the constructions, we recall the KE assumption [Dam92, HT98]. Informally, the KE assumption says that one can not compute, on input p, q, g, g^a, a pair of elements (g^r, g^{ra}) without knowing r. Essentially, this assumption claims that the only viable way of computing such a pair is by raising g and g^a to power r. This assumption can be formulated with or without independent auxiliary information. However, it does not hold with respect to dependent auxiliary information. For instance, a machine may receive $g^{r_1}, g^{r_1 a}$ as dependent auxiliary information and can output $(g^{r_1})^{r_2}, (g^{r_1 a})^{r_2}$ without knowing $r_1 * r_2$. The formal definition follows, with the convention that auxiliary information can be disregarded by removing all boxed text.

Assumption 3.3.1 (KE Assumption, [?]). *Let $\mathbb{PQGA}$ denote the distribution on (p, q, g, g^a), where p and q are uniform primes such that $p = 2q + 1$ and $|p| = n$, g is a generator for the quadratic residue group (modulo p), and a is a uniform element in $\mathbb{Z}_q^*$.*

Then, for any nonuniform PPT, A (with random coins r_A), there is another nonuniform PPT, $\mathcal{K}$, such that $\boxed{\text{for any distribution } \mathbb{Z}}$:

$$Pr[(p,q,g,g^a) \leftarrow PQGA_n,\ \boxed{z \leftarrow Z_n},\ (y_1,y_2) = A(\boxed{z},p,q,g,g^a,r_A),$$

$$x \leftarrow \mathcal{K}_A(\boxed{z},p,q,g,g^a,r_A) : y_1 = g^x \text{ or } y_2 \neq y_1^a] \leq \mu(n).$$

3.3.1.1 Extractable One-way Function

The KE and discrete-log (DL) assumptions imply that the following construction is an extractable one-way (EOW) family ensemble.

Construction 3.3.1. *Let $\mathbb{F} = \{\{f_{p,q,g,g^a}\}_{(p,q,g,g^a)\in PQGA_n}\}_{n\in\mathbb{N}}$ be a family ensemble, where*

$$f_{p,q,g,g^a}(x) = g^x, (g^a)^x$$

Specifically, the KE assumption yields extractability in a straightforward way. Moreover, by the DL assumption, $\mathbb{F}$ is one-way, where the probability is taken over the choices of the function and the input. Formally:

Definition 3.3.1 (One-way families, [DH76]). *A family ensemble, $\mathbb{F}$, where $\mathbb{F} = \{\{f_k\}_{k\in K_n}\}_{n\in\mathbb{N}}$ is one-way if for any PPT, A:*

$$Pr[k \leftarrow K_n,\ x \leftarrow U_n,\ f_k(A(f_k(x))) = f_k(x)] \leq \mu(n).$$

Theorem 3.3.1. *If Assumptions 3.3.1 (respectively with auxiliary information) is satisfied, then Construction 3.3.1 is extractable as in Definition 3.2.2 (respectively, with independent auxiliary information, as in Definition 3.2.3).*

Moreover, if Assumption 2.3.1 (respectively, Assumption 2.3.2) holds then Construction 3.3.1 is one-way (as in Definition 3.3.1) (respectively, as in Definition 2.2.1).

Proof. **Extraction:** Extraction follows immediately if we define $V_{\mathbb{F}}(x, y = (y_1, y_2)) = 1$ if and only if $g^x, (g^a)^x = y_1, y_2$.

One-wayness (as in Definition 3.3.1): For every adversary, A, that attacks this construction, let B be another adversary that contradicts the underlying assumption (Assumption 2.3.1). B receives p, q, g and y and outputs $A(p, q, g, g^a, y, y^a)$, where a is chosen uniformly by B.

One-wayness (as in Definition 2.2.1): For every adversary, A, that attacks this construction, for some p, q, g, g^a, let B be another adversary that contradicts the underlying assumption (Assumption 2.3.2). B receives p, q, g and y and outputs $A(p, q, g, g^a, y, y^a)$, where a is in the advice string of B.

□

3.3.1.2 Extractable Pseudorandom Generator

Construction 3.3.1 is pseudorandom if the DDH assumption is satisfied. This is so because the DDH assumption implies that g^a, g^x, g^{ax} is indistinguishable from g^a, g^x, g^w. Formally,

Theorem 3.3.2. *If Assumption 2.3.3 holds, then Construction 3.3.1 is a pseudorandom generator (as in Definition 2.4.2).*

Proof. Suppose for the purpose of contradiction that Construction 3.3.1 is not a family of pseudorandom generators. Then there exists a nonuniform PPT, A, such that the following is nonnegligible:

$$|Pr[(p, q, g, g^a) \leftarrow PQGA_n,\ x \leftarrow U_n,\ b \leftarrow A(f_{p,q,g,g^a}(x)) : b = 1]-$$

$$Pr[(p, q, g, g^a) \leftarrow PQGA_n,\ x \leftarrow U_n,\ b \leftarrow A(U_{|f_{p,q,g,g^a}(x)|}) : b = 1]|.$$

This implies that A breaks the DDH assumption:$p, q, g, g^a, f_{p,q,g,g^a}(y) = p, q, g, g^a, g^x, g^{ax}$ and $p, q, g^a, U_{|f_{p,q,g,g^a}(x)|} \equiv p, q, g^a, g^b, g^c$, where c is uniform. This contradicts the DDH assumption. □

3.3.1.3 Extractable Perfectly One-way Function

The final construction from the KE assumption is that of an extractable POW function. Recall from Chapter 2 that a POW function satisfies a strong notion of one-wayness, where the function hides all partial information about the input.

A starting point is the previous construction. However, it is not perfectly one-way since it reveals g^x. To fix this, we use the construction of [Can97] in which x is hidden by masking it with a uniform element r. So, the new candidate is g^{rx}, g^{rxa}. However, using the KE assumption on this construction allows us to extract rx but not x. Thus,

we add g^r, g^{ra} to the output. Formally,

Construction 3.3.2. *Let* $\mathbb{H} = \{\{H_{p,q,g,g^a}\}_{(p,q,g,g^a)\in PGGA_n}\}_{n\in\mathbb{N}}$ *be a family ensemble, where* $H_{p,q,g,g^a} : \mathbb{Z}_q^* \times R_n = \mathbb{Z}_q^* \to (QR_p)^4$ *and :*

$$H_{p,q,g,g^a}(x,r) = g^r, g^{ar}, g^{rx}, g^{arx}.$$

In the remainder of this section, we show that Construction 3.3.2 is an EPOW function. However, to prove that it is a POW function, we need a strong version of the DDH assumption as in [HT98, BP04b], where it holds for any group in some set, $\mathbb{PQG}$. This is so because we require secrecy to hold for *any* function in the family. On the other hand, the standard DDH assumption (Assumption 2.3.3) is sufficient to prove secrecy for a random function.

The DDH assumption, both the standard and the strong version, can be formalized with or without auxiliary information. Whether the assumption holds with or without auxiliary information translates directly to whether the construction is perfectly one-way with or without auxiliary information. The nonstandard version of the DDH assumption follows with auxiliary information surrounded by boxes. So, the definition without auxiliary information can be obtained by removing the boxes and their content.

Assumption 3.3.2 (Strong DDH). *Let* $\mathbb{PQG}$ *be a space of tuples* (p,q,g)*, where* p *and* q *are primes,* $p = 2q+1$*, and* g *is a generator for the quadratic residue group modulo* p*. Then, for any* $(p,q,g) \in PQG_n$*, any well-spread distribution,* $\mathbb{X}^q$*, over* $\mathbb{Z}_q^*$*,* $\boxed{\text{any uninvertible function, } F,}$ *and any nonuniform PPT,* A*:*

$$|Pr[x \leftarrow X_n^q,\ r \leftarrow \mathbb{Z}_q^*,\ \boxed{z \leftarrow F(x)},\ b \leftarrow A(p,q,g,\boxed{z}, g^x, g^r, g^{rx}) : b = 1] -$$

$$Pr[x \leftarrow X_n^q,\ r_1, r_2 \leftarrow \mathbb{Z}_q^*, \mathbb{Z}_q^*,\ \boxed{z \leftarrow F(x)},\ b \leftarrow A(p,q,g,\boxed{z}, g^x, g^{r_1}, g^{r_2}) : b = 1]| \leq \mu(n).$$

Theorem 3.3.3. *If Assumption 3.3.1 holds (respectively with independent auxiliary information), then Construction 3.3.2 is extractable, as in Definition 3.2.2(respectively, with independent auxiliary information, as in Definition 3.2.3).*

Moreover, if Assumption 3.3.2 holds (respectively, with auxiliary information), then construction 3.3.2 is computationally indistinguishable (respectively, with auxiliary information) as in Definition 2.5.5.

Proof.

Preimage extraction: If the KE assumption holds (respectively, with auxiliary information) then for any PPT, A, that outputs a valid image $(g^r, g^{ar}, g^{rx}, g^{arx})$, there are two PPT, $\mathcal{K}_1$ and $\mathcal{K}_2$, such that $\mathcal{K}_1$ extracts r and $\mathcal{K}_1$ extracts rx. Consequently, x is extractable (respectively with independent auxiliary information).

Information hiding: $\mathbb{H}$ is t-indistinguishable (respectively, with auxiliary information) if and only if $\mathbb{H}'$, where $H'_{p,q,g}(x,r) = g^r, g^{rx}$ is (respectively, with auxiliary information). For any p, q, g, g^a and any well-spread distribution, $\mathbb{X}^q$, and any PPT, A, that tries to distinguish t images of $\mathbb{H}$, there is another PPT, B, that tries to distinguish t images of $\mathbb{H}'$ with the same success rate as A. B simply raises the appropriate input to the a^{th} power and simulates A. Specifically, on input $p, q, g, \boxed{z,} y_1^1, y_1^2, ..., y_{t(n)}^1, y_{t(n)}^2$ and auxiliary input a, B runs A on $p, q, g, g^a, \boxed{z,} y_1^1, (y_1^1)^a, y_1^2, (y_1^2)^a, ..., y_{t(n)}^1, (y_{t(n)}^1)^a, y_{t(n)}^2, (y_{t(n)}^2)^a$ and outputs whatever A does.

So, for simplicity, we prove $\mathbb{H}'$ is t-indistinguishable instead. We do so by first showing that $\mathbb{H}'$ satisfies 2-indistinguishability and that 2-indistinguishability and t-indistinguishability are equivalent for this construction (where t is any polynomial).

$\mathbb{H}'$ is computationally 2-indistinguishable. At a high level, the DDH assumption implies that $H'(x), H'(x)$ is computationally indistinguishable from $H'(x), H'(U_n)$. Using the DDH assumption again, the latter distribution is computationally indistinguishable from $H'(U_n), H'(U_n)$. Formally, for any well-spread distribution, $\mathbb{X}$, $\boxed{\text{any uninvertible function, } F}$, and any nonuniform PPT, A:

$$|Pr[x \leftarrow X_n^q,\ r_1, r_2 \leftarrow \mathbb{Z}_q^*, \mathbb{Z}_q^*,\ \boxed{z \leftarrow F(x)},\ b \leftarrow A(p, q, g, \boxed{z}, g^{r_1}, g^{r_2}, g^{r_1 x}, g^{r_2 x}) : b = 1] -$$

$$Pr[x \leftarrow X_n^q,\ u_1, r_1, r_2 \leftarrow \mathbb{Z}_q^*, \mathbb{Z}_q^*, \mathbb{Z}_q^*,\ \boxed{z \leftarrow F(x)},$$

$$b \leftarrow A(p, q, g, \boxed{z}, g^{r_1}, g^{r_2}, g^{r_1 x}, g^{r_2 u_1}) : b = 1]| \leq \mu(n). \tag{3.1}$$

Otherwise, there is a distinguisher B for the DDH assumption: on input (g^x, g^{r_1}, g^z), B chooses r_2 uniformly and runs A on $p, q, g, g^{r_2}, g^{r_1}, g^{r_2 x}, g^z$. Using the same argument, we have for the same parameters as before:

$$|Pr[x \leftarrow X_n^q,\ u_1, r_1, r_2 \leftarrow \mathbb{Z}_q^*, \mathbb{Z}_q^*,\ \boxed{z \leftarrow F(x)}],$$

$$b \leftarrow A(p,q,g,\boxed{z},g^{r_1},g^{r_2},g^{r_1 x},g^{r_2 u_1}) : b = 1]-$$

$$Pr[x \leftarrow X_n^q,\ u_1,u_2,r_1,r_2 \leftarrow \mathbb{Z}_q^*,\mathbb{Z}_q^*,\mathbb{Z}_q^*,\ \boxed{z \leftarrow F(x)},$$

$$b \leftarrow A(p,q,g,\boxed{z},g^{r_1},g^{r_2},g^{r_1 u_1},g^{r_2 u_2}) : b = 1]| \leq \mu(n). \tag{3.2}$$

Eq. 3.1 and 3.2 imply that $\mathbb{H}'$ is 2-indistinguishable.

$\mathbb{H}'$ is computationally t-indistinguishable. If $\mathbb{H}'$ is 2-indistinguishable, it is t-indistinguishable for any polynomial t (respectively, with auxiliary information). We claim that given any two images, $H'_{p,q,g}(x_1,r_1), H'_{p,q,g}(x_2,r_2)$, it is possible to generate a new image, $H'_{p,q,g}(x_3,r_3)$ satisfying two conditions. First, r_3 is uniform and independent of x_1,x_2,r_1,r_2. Second, $x_3 = x_1$ if $x_1 = x_2$ and otherwise x_3 is uniform and independent of x_1,x_2,r_1,r_2,r_3. Consequently, any distinguisher, A, for t copies can be turned into a distinguisher for 2 copies that generates $t-2$ new copies and runs A.

Let G be a PPT, where on input $H'_{p,q,g}(x_1,r_1) = g^{r_1}, g^{r_1 x_1}, H'_{p,q,g}(x_2,r_2) = g^{r_2}, g^{r_2 x_2}$, G samples uniformly and independently u_1,u_2 from $\mathbb{Z}_q$, such that u_1,u_2 are not both 0, and outputs $H'_{p,q,g}(x_3,r_3) = g^{r_1 u_1 + r_2 u_2}, g^{r_1 u_1 x_1 + r_2 u_2 x_2}$. We show that the output of G satisfies the two conditions mentioned above if neither r_1 nor r_2 is 0 (the probability of either one of them being zero is negligible). If $x_1 = x_2$, then $x_3 = x_1$ and r_3 is uniform and independent of $H'_{p,q,g}(x_1,r_1), H'_{p,q,g}(x_2,r_2)$. Consequently, $H'_{p,q,g}(x_3,r_3)$ is a new image of x_1 with independent random coins. On the other hand, if $x_1 \neq x_2$, then $H'_{p,q,g}(x_3,r_3)$ is an image of independent and uniform element using independent random coins. Specifically, for any x_1,x_2,r_1,r_2, where $x_1 \neq x_2$ and $r_1 \neq 0$ and $r_2 \neq 0$, and any x_3,r_3, there is a unique pair, u_1,u_2 such that $r_3 = r_1 u_1 + r_2 u_2$ and $x_3 = (r_1 u_1 x_1 + r_2 u_2 x_2) r_3^{-1}$. Solving these two equations for u_1,u_2, we have $u_1 = (r_3 - r_3(x_3 - x_1)(x_2 - x_1)^{-1}) r_1^{-1}$ and $u_2 = r_2^{-1} r_3 (x_3 - x_1)(x_2 - x_1)^{-1}$.

Alternative proof (hybrid argument):$\mathbb{H}'$ is computationally t-indistinguishable. Suppose there exists a polynomial t and a PPT, A such that $\mathbb{H}'$ is not t-indistinguishable with respect to A. Then, by a hybrid argument, there exists an $1 \leq i \leq t$, such that the following difference is nonnegligible:

$$Advantage(A) \equiv$$

$$|Pr[x \leftarrow X_n^q,\ r_1,...,r_{t(n)},u_1,...,u_{t(n)} \leftarrow \mathbb{Z}_q^*,\ \boxed{z \leftarrow F(x)},$$

$$b \leftarrow A(p,q,g,\boxed{z},H'_{p,q,g}(x,r_1),...,H'_{p,q,g}(x,r_{i+1}),H'_{p,q,g}(u_{i+2},r_{i+2}),...,H'_{p,q,g}(u_{t(n)},r_{t(n)})):$$

$$b=1]-$$

$$Pr[x \leftarrow X_n^q,\ r_1,...,r_{t(n)},u_1,...,u_{t(n)} \leftarrow \mathbb{Z}_q^*,\ \boxed{z \leftarrow F(x)},$$

$$b \leftarrow A(p,q,g,\boxed{z},H'_{p,q,g}(x,r_1),...,H'_{p,q,g}(x,r_i),H'_{p,q,g}(u_{i+1},r_{i+1}),...,H'_{p,q,g}(u_{t(n)},r_{t(n)})):$$

$$b=1]|$$

Let B be a PPT that uses A to contradict the strong DDH assumption. B receives $p,q,g,\boxed{z}g^a,g^b,g^{ab}$. It chooses a uniform i between 1 and $t(n)$, uniformly samples $r_1,...,r_{t(n)}$ and $u_{i+2},...,u_{t(n)}$ and outputs

$$A(p,q,g,\boxed{z},(g^{r_1},(g^a)^{r_1}),...,(g^{r_i},(g^a)^{r_i}),(g^b,g^{ab}),(g^{r_{i+2}},g^{r_{i+2}u_{i+2}}),...,(g^{r_{t(n)}},g^{r_{t(n)}u_{t(n)}}).$$

By a standard hybrid argument:

$$|Pr[a \leftarrow X_n^q,\ b,c \leftarrow \mathbb{Z}_q^*,\ \boxed{z \leftarrow F(a)},b \leftarrow B(p,q,g,\boxed{z},g^a,g^b,g^{ab}):b=1]-$$

$$Pr[a \leftarrow X_n^q,\ b,c \leftarrow \mathbb{Z}_q^*,\ \boxed{z \leftarrow F(a)},b \leftarrow B(p,q,g,\boxed{z},g^a,g^b,g^c):b=1]|$$

$$> \frac{1}{t(n)}Advantage(A),$$

which is nonnegligible, contradicting the strong DDH assumption.

□

3.3.1.3.1 Verification. The verification that we have for Construction 3.3.2 is different from the usual notion. Specifically, $V_{\mathbb{H}}$ is assumed to get a as input, which is not part of the public description of $\mathbb{H}$. Specifically, $V_{\mathbb{H}}(x,(y_1,y_2,y_3,y_4),a)=1$ if and only if $y_2=(y_1)^a$,$y_4=(y_3)^a$, $y_3=(y_1)^x$ and $y_4=(y_2)^x$.

Moreover, note that this construction satisfies a form of range verification where, given a, it is easy to verify that a string is a valid image. Specifically, for any string, (y_1,y_2,y_3,y_4), if $y_2=(y_1)^a$ and $y_4=(y_3)^a$, then the image must be a valid one for some x. This is an interesting property that we use in the 3-round ZK construction. We emphasize that this property is nontrivial because an extractable function has a sparse range. Formally,

Definition 3.3.2 (Range Verification). *A family ensemble, $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$, where for some polynomial $l(n)$, for any $n \in \mathbb{N}$, and any $k, z_k \in K_n$ (z_k is auxiliary information about k), $H_k : \{0,1\}^n \times R_n \to \{0,1\}^{l(n)}$, has an efficient **range verification** if there exists a deterministic polynomial time algorithm, $V_{range(\mathbb{H})}$ such that:*

$$\forall k \in K_n,\ V_{range(\mathbb{H})}(y, z_k) = 1 \text{ iff } \exists, x, r,\ H_k(x, r) = y.$$

3.3.1.3.2 Collision resistance. It can be shown that Construction 3.3.2 is collision-free by using the fact that g^r is a permutation.

3.3.2 Constructions from the Diffie-Hellman Knowledge of Exponent Assumption

The Diffie-Hellman Knowledge of Exponent (DH-KEA) assumption is stronger than the KE assumption and implies the latter assumption [PX09]. Thus, it implies all the results in Section 3.3.1.

Informally, the DH-KEA assumption says that it is hard for any efficient adversary to compute a DDH tuple, g^a, g^b, g^{ab}, without knowing either a or b. Formally, let Gen be a group generator that takes a security parameter, k, and outputs group description, G, and an element $g \in G$.[2]

Assumption 3.3.3 (Diffie-Hellman Knowledge of Exponent Assumption, [PX09]). *There exists a PPT, Gen where for any PPT, A (with random coins r_A), there is a PPT extractor, $\mathcal{K}_A$, such that:*

$$Pr[(G, g) \leftarrow Gen(1^k),\ (A, B, C) = A(G, g, r_A),\ x \leftarrow \mathcal{K}_A(G, g, r_A) :$$

$$(\exists a, b : A = g^a, B = g^b, C = g^{ab}) \text{ and } C \neq A^x \text{ and } C \neq B^x] < \mu(n).$$

3.3.3 Constructions from the Proof of Knowledge Assumption

Lepinski [Lep02] constructs 3-round zero-knowledge *proofs* using a strong knowledge assumption, called the proof of knowledge (POK) assumption. In the context of this

[2] In [PX09], Gen generates, in addition, an upper bound on the order of g and a trapdoor for G. However, these are not needed for this assumption. For instance, the order of G is an upper bound on the order of g.

chapter, we use this assumption to construct extractable one-way and extractable perfectly one-way functions. Table 3.1 lists the results with the required assumptions.

3.3.3.1 The POK assumption

Informally, this assumption attempts to replace a Random Oracle with a hash function (or a family of such functions) in a specific 2-round proof of knowledge protocol without compromising it.[3] This is then used to construct 3-round ZK proofs for any language in NP. One of the primary usages of random oracles in this protocol is to check that the potentially-malicious prover generates a uniform string in an honest way, as specified by the protocol. So, this uniform string is designated to be the output of the Random Oracle on an input chosen by the prover (and sent to the verifier). This forces the prover to "know" something, specifically the discrete log of one of two elements. Then, this assumption comes in to replace the Random Oracle with a hash function without loosing the existence of a (non-blackbox) knowledge extractor.

The POK assumption is particular to the proof of knowledge mentioned above and may look peculiar at a first reading. Informally, the adversary is given a prime, p, a generator for $\mathbb{Z}_p^*$, an element, C, of $\mathbb{Z}_p^*$, and a hash function, h. Its task is to find pairs, $(X, Y), (W_1, Z_1), ..., (W_n, Z_n)$ such that every pair multiplies to C. Also, it has to find $B_1, ..., B_n$ where $b_1...b_n = h(X, W_1, ..., W_n)$ and $g^{B_i} \in \{W_i, Z_i\}$ if $b_i = 0$, otherwise, $g^{B_i} \in \{XW_i^{-1}, X^{-1}W_i, XZ_i^{-1}, X^{-1}Z_i\}$. The assumption then goes on to say that for some family of hash functions, every adversary should know a discrete log of X or Y if its output is valid. For clarity, we use $Test$ to denote a deterministic PPT that outputs 1 if the above conditions are satisfied and 0 otherwise. Formally,

$$Test(p, q, g, C, h, (X, Y), (W_1, Z_1), ..., (W_n, Z_n), B_1, ..., B_n) = 1$$

if and only if all the above conditions, using the input to $Test$ as parameters, are met. Even though the original assumption considers all primes, we restrict primes to safe ones (i.e., primes of the form $p = 2q + 1$, where q is prime) as in the previous section.

Assumption 3.3.4 (Proof of Knowledge (POK) Assumption, [Lep02]). *Let $\mathbb{PQGC}$ denote the uniform distribution on (p, q, g, C), where p and q are primes, $p =$*

[3] This proof of knowledge is for the case where the prover chooses (X, Y), sends them to the verifier, and proves knowledge of the discrete log of one of them.

$2q+1$, *g is a generator for the quadratic residue group,* QR_p, *and* C *is an element of* QR_p. *There exists a family of hash functions,* $\mathbb{H}$, *where for any PPT,* A *(with random coins* r_A*), there exists a PPT,* $\mathcal{K}_A$, *such that* $\boxed{\text{for any distribution } \mathbb{Z}}$:

$$Pr[(p,q,g,C) \leftarrow PQGC_n,\ h \leftarrow H_n,\ \boxed{z \leftarrow Z_n},$$

$$(X,Y),((W_1,Z_1),...,(W_n,Z_n),B_1,...,B_n) \leftarrow A(\boxed{z},p,q,g,C,h,r_A),$$

$$x \leftarrow K_A(\boxed{z},p,q,g,C,h,r_A):$$

$$Test(p,q,g,C,(X,Y),(W_1,Z_1),...,(W_n,Z_n),B_1,...,B_n)=1 \text{ and } g^x \notin \{X,Y\}] \leq \mu(n).$$

3.3.3.2 Extractable One-way Function

The POK and DL assumptions imply the existence of an extractable one-way function in a straightforward way. The function description is (p,q,g,C,h). $F_{p,q,g,C,h}$ on input x, computes g^x and Cg^{-x} and uniformly assigns X to one of the two strings and Y to the other one. Then, F chooses uniform $w_1,...,w_n$, computes the pairs $(g^{w_1},Cg^{-w_1}),...,$ (g^{w_n},Cg^{-w_n}), and assigns (W_i,Z_i) to be a random permutation on each pair. Also, F computes $b_1,...,b_n = h(X,W_1,...,W_n)$ and $B_i = w_i$ if $b_i = 0$, otherwise B_i is uniformly selected from the set $\{x-w_i,-x+w_i\}$. Finally, F outputs $(X,Y),(W_1,Z_1),...,(W_n,Z_n),B_1,...,B_n$. The POK assumption implies that this function is extractable. On the other hand, this function is one-way by the DL assumption. However, this function is not perfectly one-way because it reveals g^x. Formally,

Construction 3.3.3. *Let* $\mathbb{F} = \{\{F_{p,q,g,C,h}\}_{(p,q,g,C,h)\in PQGCH_n}\}_{n\in\mathbb{N}}$ *be a randomized family ensemble, where*

$$F_{p,q,g,C,h}(x) = (X,Y),(W_1,Z_1),...,(W_n,Z_n),B_1,...,B_n,$$

where (X,Y) *is a random permutation of* g^x, Cg^{-x}, W_i,Z_i *is a random permutation of* (g^{w_i},Cg^{w_i}), g^{w_i} *is a random element of* ZR_p^*, *and* $B_i = w_i$ *if* $h_i(X,W_1,...,W_n)=0$ *and* B_i *is a random element from* $\{x-w_i,-x+w_i\}$ *otherwise.*

We specify an efficient verifier to decide whether y is an image of x under F. $V_{\mathbb{F}}(y,x) = 1$, if and only if $Test(p,q,g,C_1,h,y) = 1$ and $g^x \in \{X,Y\}$, where y is

parsed as $(X,Y),(W_1,Z_1),...,(W_n,Z_n),B_1,...,B_n$. Note that under this definition, there are at most two (possibly distinct) preimages for each point in the range: the discrete logarithms of X and Y.

Theorem 3.3.4. *If Assumption 3.3.4 is satisfied (respectively with auxiliary information), then Construction 3.3.3 is extractable as in Definition 3.2.2 (respectively, with independent auxiliary information, as in Definition 3.2.3).*

Moreover, if the DL assumption (Assumption 2.3.1) holds, then Construction 3.3.3 is one-way (as in Definition 2.2.2).

Proof. **Extraction.** Let A be any PPT that outputs a valid image, y. By definition of $V_{\mathbb{F}}$, $Test(p,q,g,C,h,y)=1$. By Assumption 3.3.4, there is an extractor, $\mathcal{K}_A$ that outputs x such that $g^x \in \{X,Y\}$, where $y=(X,Y),(W_1,Z_1),...,(W_n,Z_n),B_1,...,B_n$. By definition, $V_{\mathbb{F}}(x,y)=1$. Consequently, x is a preimage of y.

One-wayness. Let A be any PPT that given

$$y=(X,Y),(W_1,Z_1),...,(W_n,Z_n),B_1,...,B_n$$

, outputs x such that $V_{\mathbb{F}}(x,y)=1$ with nonnegligible probability. We use A to contradict the DL assumption as follows. Let B be a PPT that receives p,q,g,D, chooses h and $a,w_1,...,w_n$ uniformly, and computes g^a and $C=Dg^a$. Then, B simulates A on $y=(X,Y),(W_1,Z_1),...,(W_n,Z_n),B_1,...,B_n$ and outputs whatever A does. Here, (X,Y) is a random permutation of (D,g^a), (W_i,Z_i) is a random permutation of (g^{w_i},Cg^{w_i}), and $B_i=w_i$ if $h_i(X,W_1,...,W_n)=0$ (h_i is the i^{th} bit of $h(.)$), otherwise B_i is uniformly chosen from $\{a-w_i,-a+w_i\}$. Since B knows the discrete log of a uniform element in $\{X,Y\}$ and A does not know which, the probability that B recovers the discrete log of D is half the probability that A recovers a preimage of y. Therefore, the latter probability should be negligible. A contradiction. □

We remark that Construction 3.3.3 is not pseudorandom because $XY=W_1Z_1=...=W_nZ_n=C$.

3.3.3.3 Extractable Perfectly One-way Function

To achieve perfect one-wayness, we use a construction similar to the one in [Can97]. Consider the following construction. $G_{p,q,g,C_1,C_2,h}$ on input x, chooses a random r, and

simply outputs $F_{p,q,g,C_1,h}(r,.), F_{p,q,g,C_2,h}(rx,.)$, where "." stands for the random coins of F. Formally,

Construction 3.3.4. *Let* $\mathbb{F} = \{\{f_{p,q,g,C,h}\}_{(p,q,g,C,h)\in PQGCH_n}\}_{n\in\mathbb{N}}$ *be the randomized family ensemble from Construction 3.3.3.*

Then, let $\mathbb{G} = \{\{G_{p,q,g,C_1,C_2,h}\}_{(p,q,g,C_1,C_2,h)\in PQGCCH_n}\}_{n\in\mathbb{N}}$ *be family ensemble defined as follows:*

$$G_{p,q,g,C_1,C_2,h}(x, r' = (r, r_1^F, r_2^F)) = F_{p,q,g,C_1,h}(r, r_1^F), F_{p,q,g,C_2,h}(rx, r_2^F),$$

where r_1^F *and* r_2^F *are the random coins for* F*.*

To verify if $y = (y_1, y_2)$ is a valid image of x under $\mathbb{G}$, the verifier, $V_{\mathbb{G}}$, accepts if $Test(p,q,g,C_1,y_1) = Test(p,q,g,C_2,y_2) = 1$ and either X_1^x or Y_1^x is in the set $\{X_2, Y_2\}$ (where $y_1 = (X_1, Y_1), ...$ and $y_2 = (X_2, Y_2), ...$).

We claim that G is extractable based on the POK assumption. Using the POK assumption on the output of the first F gives us the discrete log of either X_1 or Y_1, which we denote by r'. Likewise, the same assumption on the output of the second F, gives us the discrete log of either X_2 or Y_2, denoted by $r'x'$. Thus, x', a valid preimage, can be recovered.

Using the strong DDH assumption, we show that this construction is perfect one-way. Specifically, we show that it is 1-indistinguishable (see Definition 2.5.5). However, we don't know if this construction satisfies the more general notion of t-indistinguishability.

Theorem 3.3.5. *If Assumption 3.3.4 holds with auxiliary information, then Construction 3.3.4 is extractable with independent auxiliary information (as in Definition 3.2.3). Moreover, if Assumption 3.3.2 holds (respectively, with auxiliary information), then* $\mathbb{G}$ *is computationally* 1*-indistinguishable (respectively, with auxiliary information) as in Definition 2.5.5.*

Proof. **Extraction.** Let A be any PPT which on input (z,p,q,g,C_1,C_2,h) outputs a valid image, $y = (y_1, y_2)$. Define two new PPT, A_1 and A_2, where $A_i(z,p,q,g,C_i,h)$, receives C_{3-i} as auxiliary information, computes $A(z,p,q,g,C_1,C_2,h) = y_1, y_2$, and outputs y_i. By the POK assumption, there exists two PPT machines, $\mathcal{K}_{A_1}$ and $\mathcal{K}_{A_2}$ that compute the discrete log of either X or Y returned by the corresponding machine.

Consequently, $\mathcal{K}_A$ runs $\mathcal{K}_{A_1}$ and $\mathcal{K}_{A_2}$ to recover discrete logs, r' and $r'x'$, from the first and second message, and returns x'.

Perfect one-wayness. Computational 1-indistinguishability for $\mathbb{G}$ follows directly from the strong DDH assumption. For simplicity, we give the proof for indistinguishability without auxiliary information.

Suppose a PPT, A, breaks the perfect one-wayness of $\mathbb{G}$. Then, let B be a PPT that contradicts the strong DDH assumption. B receives $p, q, g, g^a, g^b, g^{ab}$ as input. It uniformly samples y_1, y_2, sets $C_1 = g^b Y_1$ and $C_2 = g^{ab} Y_2$, where $Y_1 = g^{y_1}$ and $Y_2 = g^{y_2}$. Also, it uniformly selects $w_1^1, ..., w_n^1, w_1^2, ..., w_n^2$, sets $W_i^j = g^{w_i^j}$ and $Z_i^j = C_j (W_i^j)^{-1}$. It randomly permutes the pairs (X_j, Y_j) and (W_i^j, Z_i^j). It also samples a hash function h, computes $b_1^j, ..., b_n^j = h(X_j, W_1^j, ..., W_n^j)$ and $B_i^j = w_i^j$ if $b_i^j = 0$, otherwise B_i^j is randomly chosen from $\{y_j - w_i^j, -y_j + w_i^j\}$. Finally, B runs A and outputs whatever A does.

$$|Pr[a \leftarrow X_n^q,\ b, c \leftarrow Z_q^*,\ b \leftarrow B(p, q, g, g^a, g^b, g^{ab} : b = 1] -$$

$$Pr[a \leftarrow X_n^q,\ b, c \leftarrow Z_q^*,\ b \leftarrow B(p, q, g, g^a, g^b, g^c : b = 1]| \geq$$

$$Pr[x \leftarrow X_n^q,\ r \leftarrow R_G,\ b \leftarrow A(G_{p,q,g,C_1,C_2,h}(x, r)) : b = 1] -$$

$$Pr[x \leftarrow Z_q^*,\ r \leftarrow R_G,\ b \leftarrow A(G_{p,q,g,C_1,C_2,h}(x, r)) : b = 1]| > \mu(n).$$

This contradicts the DDH assumption. □

Range verification. Observe that $\mathbb{G}$ satisfies range verification (as in Definition 3.3.2) because $Test(p, q, g, C_1, y_1) = Test(p, q, g, C_2, y_2) = 1$ if and only if $y = (y_1, y_2)$ belongs to the range of $G_{p,q,g,C_1,C_2,h}$. Moreover, this range verification does not require any private information about the function $G_{p,q,g,C_1,C_2,h}$.

Collision Resistance. We show that $\mathbb{G}$ satisfies collision resistance if both the POK and DDH assumptions hold against nonuniform adversaries. Nonuniformity is needed because collision resistance is defined against nonuniform machines. Formally,

Theorem 3.3.6. *If Assumptions 3.3.4 and 2.3.3 hold against nonuniform PPT adversaries, then Construction 3.3.4 is collision resistant (as in Definition 2.5.2).*

Proof. Suppose a nonuniform PPT, A, on input p, q, g, C_1, C_2, h, outputs a collision, that is, (y, a, b) where $V_{\mathbb{G}}(y, a) = V_{\mathbb{G}}(y, b) = 1$, with nonnegligible probability. We do a case

by case analysis to obtain a contradiction. For clarity, we focus on the (X_1, Y_1) and (X_2, Y_2) part of y and denote by x_i (respectively, y_i) the discrete log of X_i (respectively, Y_i). Since both a and b are valid preimages of y, there are three possible cases:

1. $X_1^a = X_2$ and $X_1^b = Y_2$ (the same analysis holds for the symmetric cases, $X_1^b = X_2$ and $X_1^a = Y_2$, etc). By the POK assumption, we can recover either x_2 or y_2. Suppose we recover x_2, then given the output of A, we can compute $y_2 = x_1 b = x_2 a^{-1} b$. However, we can now recover the discrete log of C_2, which is $x_2 + y_2$, contradicting the DL assumption.[4] Specifically, a PPT, B, on input p, q, g, C_2, generates C_1 and h, runs A on p, q, g, C_1, C_2, h and computes $x_2 + y_2$ with nonnegligible probability as described above.

2. $X_1^a = X_2$ and $Y_1^b = X_2$ (the same analysis holds for the symmetric cases and is omitted here). By the POK assumption, we can recover either x_1 or y_1. Suppose we recover x_1. So, we can compute $y_1 = x_2 b^{-1} = x_1 a b^{-1}$. Thus, we recover the discrete log of C_1, contradicting the DL assumption.

3. $X_1^a = X_2$ and $Y_1^b = Y_2$ (the same analysis holds for the symmetric cases and is omitted here).

   (a) If, by the POK assumption, we recover (x_1, y_2) or (y_1, x_2), then we can recover (x_2, y_2), to contradict the DL assumption on C_2.

   (b) Suppose we recover (x_1, x_2) by the POK assumption (the symmetric case of y_1, y_2 has the same analysis). Denote by c_i the discrete log of C_i. Then, we obtain the following relationship between c_1 and c_2: $c_2 = bc_1 + x_1(a - b)$. We use the DDH assumption to show that this case is also not possible. On input, g^u, g^v, g^w, where u and v are uniform but w can be either uv or uniform, compute $A(p, q, g, g^u, g^v, h)$ and $A(p, q, g, g^u, g^w, h)$. If w is uniform, then A outputs a collision on both runs (with nonnegligible probability). By the DDH assumption, the same can be said when $w = uv$. In the latter case, A outputs, among other things, $(X_1, Y_1), (X_2, Y_2), a, b$ on the first run and $(X_1', Y_1'), (X_2', Y_2'), a', b'$ on the second. We then have two relations in two unknowns:

[4] The DL assumption is implied by the DDH assumption.

$v = bu + x_1(a - b)$ and $w = uv = b'u + x'_1(a' - b')$.

Solving for u contradicts the DL assumption on g^u.

□

3.4 The Relationship Between Extractable Functions and NIZK proofs of knowledge

Superficially, extractable POW functions and NIZK proofs of knowledge seem to satisfy very similar knowledge requirements. However, NIZK proofs of knowledge are weaker in two aspects. First, they require the extractor to work with the help of auxiliary information about the common reference string (CRS). On the other hand, EPOW functions require extractors to work given the view of the adversary *without any auxiliary information*. Second, secrecy of NIZK (zero-knowledge) holds over the choices of the CRS while the secrecy of EPOW functions holds for any function in the family. We show that if we weaken the secrecy requirement of EPOW functions to hold for a uniformly sampled function and strengthen the knowledge requirement of NIZK, we get an equivalence. Specifically, we show that the existence of "strong" NIZK proofs of knowledge is equivalent to the existence of extractable ("weak" POW) functions, where extraction is with independent auxiliary information. Moreover, the existence of "stronger" NIZK proofs of knowledge implies extractable weak POW functions, where extraction is with *dependent* auxiliary information. However, we do not know if the implication in the reverse direction is true.

Traditionally, NIZK arguments ask for a universal blackbox extractor that can recover a witness from the proof and some auxiliary information about the CRS. However, we strengthen this notion in one respect and relax it in another. Specifically, we require that the extractor succeed without additional information about the CRS. On the other hand, we allow the extractor to depend on the prover, and we also give it access to the prover's private input. According to this notion, the extractor has to be nonblackbox and consequently non-universal. Such a definition captures the computational notion of knowledge more accurately than the original one as it clearly demonstrates that the prover knows a witness by efficiently extracting it from the view of the prover. Formally,

Definition 3.4.1 (Strong $t-$proofs of knowledge). *Let $\mathcal{P} = (P, V)$ be a noninteractive zero-knowledge argument system in the CRS model (as in Definition 2.10.1) for an NP relation R_L. Then, it is called a* ***strong t-proof of knowledge*** *if for every PPT, A (with randomness r_A), there exists a PPT, $\mathcal{K}_A$, such that for any $(x_1, w_1), ..., (x_{t(n)}, w_{t(n)}) \in R_L$ (where $n = |x_1| = ... = |x_{t(n)}|$), and any auxiliary information z:*

$$Pr[\sigma \leftarrow U_n,\ \pi_1, ..., \pi_{t(n)} \leftarrow P(x_1, w_1, \sigma), ..., P(x_{t(n)}, w_{t(n)}, \sigma),$$

$$(x, \pi) = A(z, x_1, ..., x_{t(n)}, \pi_1, ..., \pi_{t(n)}, \sigma, r_A),\ w \leftarrow \mathcal{K}_A(z, x_1, ..., x_{t(n)}, \pi_1, ..., \pi_{t(n)}, \sigma, r_A) :$$

$$(x, w) \in R_L \text{ or } (\exists i, (x, \pi) = (x_i, \pi_i)) \text{ or } V(x, \pi, \sigma)] \neq 1 > 1 - \mu(n).$$

$\mathcal{P}$ is called a strong proof of knowledge if it is a t-proof of knowledge for every polynomial t.

Observe that Definition 3.4.1 requires not only that it is hard to generate an acceptable proof without knowing a witness but also that it is hard to find a new theorem for which one of the given proofs applies.

For the results in this section, we relax the secrecy requirement on POW functions so that it holds over the choice of the function, i.e., weak POW functions as defined in [CMR98]. Formally,

Definition 3.4.2 (weak t-Indistinguishability). *Let F be any (possibly probabilistic) uninvertible function. A verifiable family ensemble $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$, where $H_k : \{0,1\}^n \times R_n \rightarrow \{0,1\}^{l(n)}$ for some polynomial l, is called* ***weakly t-indistinguishable*** *with auxiliary input F if for any well-spread distribution, $\mathbb{X} = \{X_n\}_{n\in\mathbb{N}}$ and any PPT A:*

$$|Pr[k \leftarrow K_n,\ x \leftarrow X_n,\ \boxed{z \leftarrow F(x)},\ (r_1, ..., r_t) \leftarrow (R_n, ..., R_n) :$$

$$A(k, \boxed{z}, H_k(x, r_1), ..., H_k(x, r_t)) = 1] \quad -$$

$$Pr[k \leftarrow K_n,\ x \leftarrow X_n,\ (u_1, ..., u_t) \leftarrow (U_n, ..., U_n),\ \boxed{z \leftarrow F(x)},\ (r_1, ..., r_t) \leftarrow (R_n, ..., R_n) :$$

$$A(k, \boxed{z}, H_k(u_1, r_1), ..., H_k(u_t, r_t)) = 1]| \leq \mu(n).$$

If $\mathbb{H}$ is weak t-indistinguishable with any auxiliary input F then it is called weak t-indistinguishable with auxiliary input. Moreover, if it is weak t-indistinguishable

with auxiliary input *for any polynomial t, then it is called weak indistinguishable* *with auxiliary input*.

We show that the existence of NIZK strong 0-proofs of knowledge is equivalent to that of EPOW functions *if we consider independent auxiliary information only.* Moreover, if we consider dependent auxiliary information, then we show that NIZK strong t-proofs of knowledge are sufficient for constructing extractable weakly-indistinguishable POW functions. However, we do not know if the implication holds in the reverse direction. Formally, we have the following theorem.

Theorem 3.4.1. *If weak t-indistinguishable POW functions exist (as in Definition 3.4.2) and NIZK strong 0-proofs of knowledge (respectively, NIZK strong proofs of knowledge) exist (as in Definition 3.4.1), then extractable weak t-indistinguishable POW functions exist as well, as in Definitions 3.4.2 and 3.2.3 (respectively, as in Definitions 3.4.2 and 3.2.5).*

Moreover, if extractable weakly indistinguishable POW functions exist (as in Definitions 3.4.2 and 3.2.3) and semantically secure encryption and noninteractive witness-indistinguishable proofs exist (as in Definitions 2.7.1 and 2.11.1), then NIZK strong 0-proofs of knowledge exist (as in Definition 3.4.1).

Proof. We prove each direction by presenting a construction and then analyzing its security.

From NIZK to EPOW functions. We convert any weak POW function to a weak EPOW function by appending a proof of preimage knowledge to the output of the original function. Formally, let $\mathbb{H}$ be any weak POW. Let $L_k = \{y : \exists x, r, y = H_k(x, r)\}$ and $\mathcal{P} =< P(y, x, r, \sigma), V(y, ., \sigma) >$ be a NIZK strong proof of knowledge for L_k, where σ is the CRS. Let r_P be the random coins for P. Then, the following is an EPOW family ensemble:

$$H'_{k'=(k,\sigma)}(x, r' = (r, r_P)) = H_k(x, r), P(H_k(x, r), x, r, \sigma, r_P).$$

Verification. To verify that $y = (y_1, y_2)$ is a valid image of x, check that y_1 is a valid image of x under $\mathbb{H}$ and y_2 is an acceptable proof of preimage knowledge for y_1. Formally, $V_{\mathbb{H}'}(x, (y_1, y_2)) = 1$ if and only if $V_{\mathbb{H}}(x, y_1) = 1$ and $V(y_1, y_2, \sigma) = 1$ (the last V is the verifier for the NIZK proof).

Range verification. Observe that $\mathbb{H}'$ has a range verifier if $\mathbb{H}$ has one (with auxiliary information z_k). Specifically, $V_{range(\mathbb{H}')}(y = (y_1, y_2), z_k) = 1$ if and only if $V_{range(\mathbb{H})}(y_1, z_k) = 1$ and $V(y_1, y_2, \sigma) = 1$.

Collision resistance. Collision resistance follows directly from collision resistance on $\mathbb{H}$.

Preimage extraction. It is possible due to witness extraction of the NIZK argument. In more detail, let A be an adversary that receives the following as input

$$k' = (k, \sigma), z, H'_{k'}(x_1, r_1^{x_1}), ..., H'_{k'}(x_{t_1(n)}, r_{t_2(n)}^{x_{t_1(n)}})$$

and outputs a new image y (for any polynomials t_1 and t_2, any $x_1, ..., x_{t_1(n)}$, any auxiliary information z about $x_1, ..., x_{t_1(n)}$). By construction, $y = (y_1, y_2)$ is a new theorem/proof pair for $(L_k, \mathcal{P})$ (where y_1 is the theorem and y_2 is the proof). Thus, by witness extraction (Definition 3.4.1), there exists a $\mathcal{K}_A$ that computes a witness $w = (x, r)$ for y_1, i.e., $H_k(x, r) = y_1$. By completeness of both the NIZK proof of knowledge and verification of $\mathbb{H}$, $V_{\mathbb{H}'}(x, y) = 1$. Let $\mathcal{K}'_A$ be same as $\mathcal{K}_A$ except that it outputs x instead of $w = (x, r)$. Then, $\mathcal{K}'_A$ is preimage extractor for A and $\mathbb{H}'$. (To get the proof for the special case of extraction with independent auxiliary information, set t_1 and t_2 to 0.)

Information hiding. We claim that a t-sequence of images under $\mathbb{H}'$ is indistinguishable from images of uniform strings. Suppose, for the purpose of contradiction, that this is not the case. We then show that $\mathbb{H}$ is not weakly t-indistinguishable. Formally, suppose, for the purpose of contradiction, there exists a well-spread distribution, $\mathbb{X}$, auxiliary information, F, polynomials, p and t, and a PPT, A', such that for infinitely many n:

$$|Pr[k' \leftarrow K'_n,\ x \leftarrow X_n,\ z \leftarrow F(x),\ (r_1, ..., r_t) \leftarrow (R'_n, ..., R'_n) :$$

$$A'(k', z, H'_{k'}(x, r_1), ..., H'_{k'}(x, r_t)) = 1] \quad -$$

$$Pr[k' \leftarrow K'_n,\ x \leftarrow X_n,\ (u_1, ..., u_t) \leftarrow (U_n, ..., U_n),\ z \leftarrow F(x),\ (r_1, ..., r_t) \leftarrow (R'_n, ..., R'_n) :$$

$$A'(k', z, H'_{k'}(u_1, r_1), ..., H'_{k'}(u_t, r_t)) = 1]| \geq \frac{1}{p(n)}. \tag{3.3}$$

We define A to be a PPT that uses A' to break the indistinguishability property of $\mathbb{H}$. A receives a sequence of images under $\mathbb{H}$, it runs the ZK simulator, $S = (S_1, S_2)$, to

convert images under $\mathbb{H}$ to images under $\mathbb{H}'$, and then runs A' on them. Formally,

$$A(k, z, y_1, ..., y_t) = A'(k', z, (y_1, S_2(\sigma, aux(\sigma), y_1)), ..., (y_t, S_2(\sigma, aux(\sigma), y_t))),$$

where $S_1(k) = (\sigma, aux(\sigma))$ and $k' = (k, \sigma)$.

By the zero-knowledge property, we have

$$|Pr[k' \leftarrow K'_n,\ x \leftarrow X_n,\ z \leftarrow F(x),\ (r_1, ..., r_t) \leftarrow (R'_n, ..., R'_n) :$$

$$A'(k', z, H'_{k'}(x, r_1), ..., H'_{k'}(x, r_t)) = 1] -$$

$$Pr[k \leftarrow K_n,\ x \leftarrow X_n,\ z \leftarrow F(x),\ (r_1, ..., r_t) \leftarrow (R_n, ..., R_n) :$$

$$A(k, z, H_k(x, r_1), ..., H_k(x, r_t)) = 1]| \leq \mu(n). \tag{3.4}$$

$$|Pr[k' \leftarrow K'_n,\ x \leftarrow X_n,\ (u_1, ..., u_t) \leftarrow (U_n, ..., U_n),\ z \leftarrow F(x),\ (r_1, ..., r_t) \leftarrow (R'_n, ..., R'_n) :$$

$$A'(k', z, H'_{k'}(u_1, r_1), ..., H'_{k'}(u_t, r_t)) = 1] -$$

$$Pr[k \leftarrow K_n,\ x \leftarrow X_n,\ (u_1, ..., u_t) \leftarrow (U_n, ..., U_n),\ z \leftarrow F(x),\ (r_1, ..., r_t) \leftarrow (R_n, ..., R_n) :$$

$$A(k, \boxed{z}, H_k(u_1, r_1), ..., H_k(u_t, r_t)) = 1]| \leq \mu(n). \tag{3.5}$$

Combining Eq. 3.3, 3.4, and 3.5 contradicts the assumption that $\mathbb{H}$ satisfies weak t-indistinguishability.

From EPOW functions to NIZK.

Let $\mathbb{H}$ be an extractable (with independent auxiliary information) weak POW function and (G, E, D) be any public-key encryption scheme. Also, let L' be any NP-language with relation $R_{L'}$ and $\mathcal{P}' = (P', V')$ be a noninteractive witness indistinguishable proof for the language:

$$L_{k, pk_1, pk_2} = \{((x, y_1, y_2), (y_3, y_4)) :$$

$$\exists w, v, r_2 : (x, w) \in R_{L'} \text{ and } V_{\mathbb{H}}((w, v), y_1) = 1 \text{ and } E_{pk_1}((w, v), r_2) = y_2$$

$$\text{or } \exists u, r_3, r_4 : E_{pk_1}(u, r_3) = y_3 \text{ and } E_{pk_2}(u, r_4) = y_4\}$$

Then, the NIZK strong proof of knowledge is defined as follows:

1. **CRS.** $\sigma = (k, pk_1, pk_2, y_3 = E_{pk_1}(u, r_3))$, where k is a randomly generated key for $\mathbb{H}$, pk_1 and pk_2 are two randomly generated public keys for (G, E, D), and u and r_3 are uniformly generated strings.

2. **The Prover.**
   input: $x, w, \sigma = (k, pk_1, pk_2, y_3)$

   v, u', r_1, r_2, r_4 are sampled uniformly;
   $y_1 = H_k((w, v), r_1)$;
   $y_2 = E_{pk_1}((w, v), r_2)$;
   $y_4 = E_{pk_2}(u', r_4)$;
   return $y_1, y_2, y_4, P'((x, y_1, y_2, y_3, y_4), (w, v, r_2))$;

   Algorithm 3.4.1: P

3. **The Verifier.**
   input: $x, \pi = (y_1, y_2, y_4, \pi'), \sigma = (k, pk_1, pk_2, y_3)$

   return $V'((x, y_1, y_2, y_3, y_4), \pi')$;

   Algorithm 3.4.2: V

Completeness. For any $(x, w) \in R_{L'}$, $\sigma = (k, pk_1, pk_2, y_3)$, and any v, u', r_1, r_2, r_4 (chosen by P), we have $(x, y_1, y_2, y_3, y_4), (w, v, r_2) \in R_{L_{k,pk_1,pk_2}}$. Thus, completeness follows directly from completeness on the WI proof, $\mathcal{P}'$.

Soundness. If x is not in L' then the first tuple of a L_{k,pk_1,pk_2} statement, i.e., the tuple (x, y_1, y_2), can not be true. Moreover, without auxiliary information about the CRS, we show that no adversary can generate a true second tuple, (y_3, y_4). Consequently, soundness follows by the soundness property of $\mathcal{P}'$.

Formally, let $\hat{P}$ be any PPT. Then,

$$Pr[\sigma \leftarrow \Sigma_n,\ (x, \pi = (y_1, y_2, y_4, \pi') \leftarrow \hat{P}(\sigma) : V(x, \pi, \sigma) = 1 \text{ and } x \notin L'] \leq$$

$$Pr[\sigma \leftarrow \Sigma_n,\ (x, \pi = (y_1, y_2, y_4, \pi')) \leftarrow \hat{P}(\sigma) :$$

$$\exists u, r_3, r_4 : E_{pk_1}(u, r_3) = y_3 \text{ and } E_{pk_2}(u, r_4) = y_4] \ + \tag{3.6}$$

$$Pr[\sigma \leftarrow \Sigma_n,\ (x, \pi) \leftarrow \hat{P}(\sigma) :$$

$$V(x,\pi,\sigma) = 1 \text{ and } x \notin L' | \forall u, r_3, r_4 : E_{pk_1}(u, r_3) \neq y_3 \text{ or } E_{pk_2}(u, r_4) \neq y_4] \tag{3.7}$$

$$\leq \mu(n).$$

We need to show that both Eq. 3.6 and 3.7 are negligible.

Eq. 3.6 is negligible due to semantic security of (G, E, D). Formally, suppose that Eq. 3.6 is not negligible. Let A be the following PPT that receives auxiliary information x.

input: $x, pk_1, E_{pk_1}(u, r_3)$

1 $k \leftarrow K_n$;

2 $(pk_2, sk_2) \leftarrow G(1^n)$;

3 $\sigma = k, pk_1, pk_2, E_{pk_1}(u, r_3)$;

4 $y_1, y_2, y_4, \pi' \leftarrow \hat{P}(x, \sigma)$;

5 **return** $D_{sk_2}(y_4)$;

Algorithm 3.4.3: A

We have:

$$Pr[(pk, sk) \leftarrow G(1^n),\ u \leftarrow U_n,\ c \leftarrow E_{pk}(u),\ u' \leftarrow A(pk, x, c) : u' = u]$$

is nonnegligible. This contradicts the secrecy of the encryption scheme.

The event in Eq. 3.7 implies that $(x, y_1, y_2, y_3, y_4) \notin L_{k,pk_1,pk_2}$. The first statement, i.e., the tuple (x, y_1, y_2), is not true because $x \notin L'$. Moreover, by the condition in Eq. 3.7, (y_3, y_4) is also not true. Thus, Eq. 3.7 is negligible by the soundness property of $\mathcal{P}'$.

Witness Extraction. As we show in proving soundness, if a prover does not have auxiliary information about the CRS, the proof is not accepted unless the first statement is true. Consequently, the prover computes a valid image of the witness (concatenated with a uniform string). By preimage extraction, this witness can be extracted.

Formally, let $\hat{P}$ be any PPT that receives some auxiliary information, z, and σ, and produces a new theorem/proof pair x, π. We construct a PPT, A that produces a new

image for $\mathbb{H}$ as follows:

input: $k, z' = (z, pk_1, pk_2, E_{pk_1}(u, r_3))$

1 $(x, y_1, y_2, y_4, \pi') \leftarrow \hat{P}(z, \sigma = (k, pk_1, pk_2, E_{pk_1}(u, r_3)))$;

2 **return** y_1;

Algorithm 3.4.4: A

By preimage extraction (Definition 3.2.3), we have a PPT, $\mathcal{K}_A$ that extracts a preimage, (w', v'), of y_2. Now, we use collision resistance to show that w' is a witness for x. Observe that if V accepts a theorem/proof, $(x, \pi = (x, y_1, y_2, \pi'))$, and taking into account that Eq 3.6 is negligible, then the exists a w, v, r_2 such that $(x, w) \in R_{L'}$, $V_{\mathbb{H}}((w, v), y_1) = 1$, and $E_{pk_1}((w, v), r_2) = y_2$. Thus, by collision resistance, $w' = w$ (otherwise, an adversary can compute a collision, (w, v) and (w', v') for y_1 by selecting (pk_1, sk_1), simulating the whole experiment, and then using D_{sk_1} to recover (w, v) and $\mathcal{K}_A$ to recover (w', v')).

Zero Knowledge. The simulator, S, simply receives the plaintext and randomness used in computing y_3, as auxiliary information about the CRS. Thus, S can fake a proof for any statement x by running the witness indistinguishable prover using a witness for the second statement. By the secrecy of $\mathbb{H}$, the encryption scheme, and witness indistinguishability, this simulation is indistinguishable from a real proof. Formally,

input: $x, \sigma, aux(\sigma) = (u, r_3)$

1 $(w', v') \leftarrow U_n$;

2 $r_4 \leftarrow R_n$;

3 $y_1 \leftarrow H_k((w', v'))$;

4 $y_2 \leftarrow E_{pk_1}((w', v'))$;

5 $y_4 = E_{pk_2}(u, r_4)$;

6 **return** $y_1, y_2, y_3, P'((x, y_1, y_2, y_3, y_4), (u, r_3, r_4))$;

Algorithm 3.4.5: S

For clarity, we prove the special case where one simulated proof is indistinguishable from a real proof. Proof of the general case is similar.

By semantic security on (G, E, D), and then by 1-indistinguishability on $\mathbb{H}$, we have for any $(x, w) \in R_{L'}$ (that may depend on σ) and PPT, A:

$$|Pr[(pk_1, sk_1) \leftarrow G(1^n),\ k \leftarrow K_n,\ v \leftarrow U_n,\ y_1 \leftarrow H_k(w, v),\ y_2 \leftarrow E_{pk_1}(w, v),$$

$$b \leftarrow A(k, pk_1, x, y_1, y_2) : b = 1]-$$

$$Pr[(pk_1, sk_1) \leftarrow G(1^n),\ k \leftarrow K_n,\ u \leftarrow U_n,\ y_1 \leftarrow H_k(u),\ y_2 \leftarrow E_{pk_1}(u),$$

$$b \leftarrow A(k, pk_1, x, y_1, y_2) : b = 1]| \leq \mu(n).$$

Consequently, we have for any $(x, w) \in R_{L'}$ (that may depend on σ) and any PPT, A:

$$|Pr[b \leftarrow A(x, S(x, \sigma, aux(\sigma)), \sigma) : b = 1]-$$

$$Pr[(pk_1, sk_1), (pk_2, sk_2) \leftarrow G(1^n), G(1^n),\ k \leftarrow K_n,\ v, u \leftarrow U_n, U_n,$$

$$y_1 \leftarrow H_k(w, v),\ y_2 \leftarrow E_{pk_1}(w, v),\ r_3, r_4 \leftarrow U_n, U_n,\ y_3 = E_{pk_1}(u, r_3), y_4 = E_{pk_2}(u, r_4),$$

$$b \leftarrow A(x, (y_1, y_2, y_4, P'((x, y_1, y_2, y_3, y_4), (u, r_3, r_4)), \sigma)) : b = 1]| \leq \mu(n) \tag{3.8}$$

From Eq. 3.8 and witness indistinguishability, we have for any $(x, w) \in R_{L'}$ and any PPT, A:

$$|Pr[b \leftarrow A(x, S(x, \sigma, aux(\sigma)), \sigma) : b = 1]-$$

$$Pr[(pk_1, sk_1), (pk_2, sk_2) \leftarrow G(1^n), G(1^n),\ k \leftarrow K_n,\ v, u \leftarrow U_n, U_n,$$

$$y_1 \leftarrow H_k(w, v),\ r_2 \leftarrow U_n,\ y_2 = E_{pk_1}((w, v), r_2),\ y_3, y_4 \leftarrow E_{pk_1}(u), E_{pk_2}(u),$$

$$b \leftarrow A(x, (y_1, y_2, y_4, P'((x, y_1, y_2, y_3, y_4), (w, v, r_2)), \sigma)) : b = 1]| \leq \mu(n) \tag{3.9}$$

From Eq. 3.9 and semantic security on (G, E, D), we get for any $(x, w) \in R_{L'}$ (that may depend on σ) and any PPT, A:

$$|Pr[b \leftarrow A(x, S(x, \sigma, aux(\sigma)), \sigma) : b = 1] - Pr[b \leftarrow A(x, P(x, w, \sigma), \sigma) : b = 1]| \leq \mu(n).$$

□

Chapter 4

Interactively Extractable Functions

Summary: We introduce and formalize *another* notion of computational knowledge, called **interactively-extractable functions**, and give several constructions.

Informally, an interactively-extractable function is a *probabilistic* function which guarantees that any machine that produces *"many"* points with the same preimage, knows this preimage. The "many images" requirement is captured via a 3-round Arthur-Merlin game with a challenger. That is, this game provides evidence that a machine is capable of producing many points with the same preimage. Following the similar notion of noninteractively-extractable functions (Chapter 3), knowledge is captured via the existence of an efficient extractor that can play the role of the challenger in the 3-round game to recover a preimage.

As in Chapter 3, we formalize this notion in several models. We consider extraction for a single function and extraction for a family of functions. We also consider models with and without auxiliary information. Moreover, we study *blackbox* extraction, i.e., the extractor has only blackbox access to the adversary.

After formulating this notion, we present several constructions. The constructions satisfy two properties. The first one is knowledge extraction, specifically, interactive extraction for a single function in the presence of

This chapter is based on the paper [CD08a], which is a joint work with Ran Canetti. Note that [CD08a] contains some additional results that do not appear in this chapter.

(dependent) auxiliary information. The second property is a computational-hardness property and can be one-wayness or perfect one-wayness. All of these constructions are based on hardness assumptions, *without any knowledge assumptions*. Finally, we present a construction of a related notion from Σ-protocols (see [Blu86, CDN01] and Definition 2.12.1).

4.1 Introduction

Chapter 3 introduces the notion of noninteractively-extractable functions. These are functions for which any machine that computes a *single* point in the range, knows a corresponding image. As we discussed in Chapter 3, a major disadvantage of this notion is that all known constructions are based on strong knowledge assumptions that are not efficiently-falsifiable [Nao03].

We relax the notion of extraction so that nontrivial constructions can be realized from computational hardness assumptions. The relaxed notion requires the adversary to output more than a single image. Naturally, when we require an adversary to output many distinct images with a common preimage, we are referring to *probabilistic* functions. Recall, a probabilistic function takes two inputs x and r, where x is referred to as the actual input and r as the random coins of the function. Demanding the adversary to produce two, three, or a polynomial number of distinct images with a common preimage does not weaken the notion substantially. In fact, one can define a new function that is the concatenation of two, three, or a polynomial number of distinct images under the original function. The new function is noninteractively extractable if and only if the original function is extractable against adversaries producing the required number of images. Consequently, the relaxed notion requires output of many more images. In fact, it requires adversaries to produce a polynomial fraction of all possible images. As each input can have an exponential number of images, one for each distinct r, it can be immediately realized that such a requirement can not be met by efficient machines.

The interactive model. The workaround for this problem is to formulate this notion in a different model, namely, the interactive model. In this model, the adversary receives a challenge, which is a uniformly sampled r, and has to produce a new image of the same input but with r as the random coins of the function. Now, it is possible to

realize a requirement *similar* to the one mentioned at the end of the previous paragraph. The adversary has to send an initial image in the first stage. This image serves as a "commitment" to the preimage x (more on this later). In the second stage, the adversary receives a uniformly sampled r. The adversary then responds with a new image of x using r as random coins for the function. Note that if the adversary produces values with a common preimage with a noticeable probability (where probability is taken over r as well), then it effectively produces a polynomial fraction of all images of x. In other words, even though an efficient machine can not write down a polynomial fraction of an exponential numbers of images, it can output any image that belongs to this fraction if asked to.

In more detail, interactive extraction requires the adversary to engage in a 3-round game with a challenger. The adversary, A, sends, in the first round, a point, $y_0 = f(x, r_0)$, where x and r_0 are chosen by A. The challenger responds with random coins, r_1, in the second round, and A has to send back $y_1 = f(x, r_1)$ (see Figure 4.1). In this setting, consistency means that y_0 and y_1 have a common preimage x. Interactive extraction means if the adversary is able to answer *consistently*, then it knows a common preimage. As in the noninteractive case, this form of knowledge is captured computationally by the existence of an extractor that recovers a preimage from the private input of the adversary.

In this chapter, extraction refers to the notion described above, while we refer to the corresponding notion of Chapter 3 as *noninteractive* extraction.

On efficient verification. As in the case of noninteractive extraction, we emphasize that no efficient verification of consistency is assumed to occur. The knowledge requirement states that *if the adversary is consistent*, it must know a preimage. In fact, in some cases such as perfect one-wayness, assuming efficient verification contradicts the very hardness property we seek.

The significance of the first message. The first image, y_0, that the adversary sends, plays an important role as a binding message. In other words, consistency forces any subsequent image to share a preimage of y_0. If we remove the first message, the new game fails to capture the requirement on the adversary to produce a polynomial fraction of all images of a particular point. For instance, suppose without loss of generality that the input and randomness domain are identical. Then, it is conceivable that an adversary

matches each r with a different input x. Effectively, this adversary produces only a single image per input.

4.1.1 Our Work

This chapter is devoted to formulating and constructing interactively-extractable functions.

4.1.1.1 Formulating Extraction

The general format of a definition of extraction is similar to the one in Chapter 3: for any efficient adversary, A, that plays the 3-round game described above, there is a corresponding extractor that recovers a preimage from the private input of A.

There are *six* variants of this definition depending on three major criteria. First, extraction can be required for any function in the family or for a uniformly chosen one. Second, extraction may be in the absence or presence of independent or dependent auxiliary information. Third, extraction can be blackbox or nonblackbox.

Blackbox extraction. A *noninteractive*, nontrivial extractor has to depend on the machine generating the output and has to have access to its private input. Otherwise, the function can not be one-way. On the other hand, interactive extraction permit blackbox extractors. A blackbox extractor can take the role of the challenger in the 3-round game and rewind the game to any stage. However, it does not know the particular adversarial strategy it is communicating with nor can it read the private tape of the adversary. Consequently, this blackbox extractor is universal in the sense that there is a fixed extractor capable of computing a preimage *for any efficient adversarial strategy*. Jumping ahead, the constructions described below satisfy the best of all three criteria, that is blackbox extraction with auxiliary information for any single function from a family.

Extraction error. Noninteractive extraction requires the extractor to succeed with overwhelming probability. That is, the failure error is negligible. For interactive extraction, this notion is not known to be realizable. Instead, current constructions guarantee the extraction error to be arbitrary small but noticeable. In other words, for every polynomial, p, there is an extractor, that depends on p, and fails with probability at most $\frac{1}{p}$. In Chapter 5, we study extraction with negligible error.

On the number of challenges. The actual set of definitions require the adversary to answer n challenges instead of one. It is possible to keep the requirement to one challenge. In this case, we can easily transform any construction that satisfies the former into a construction that satisfies the latter notion by concatenating n images from the first construction. However, for efficiency, we keep the former construction as is and relax the notion to allow for n challenges. In Chapter 5, we study and realize extraction against a single challenge.

Towards more general definitions. One can directly generalize the notion of 3-round interactive extraction into a parameterized definition for t-rounds. From a different angle, general t-round games have a sequential flavor in that a challenger sends a challenge r and the adversary responds with the corresponding image. This process is repeated sequentially an appropriate number of times. On the other hand, the 3-round game as described in this chapter is parallel in nature in that the challenger sends all of its challenges once and for all. Further work and constructions that satisfy the general definitions remain the topic for future work. Consequently, we relegated these definitions to Appendix A.

4.1.1.2 Constructions

We present three constructions of extractable functions. All three of them are blackbox extractable with auxiliary information for any function taken from a family. However, the first one is one-way, the second one is perfectly one-way and the last one is perfectly one-way with auxiliary information.

Extractable one-way functions. The one-way construction utilizes a special form of verifiable secret sharing (VSS) schemes [CGMA85, Fel87]. Informally, a secret-sharing (SS) scheme [Bla79, Sha79] allows a dealer to split a secret into a number of shares such that it is not possible to recover the secret unless a sufficient quorum of shares is present. VSS requires, in addition, efficient verification of the share to guarantee validity and uniqueness of the secret (see Section 4.3.1 for more detail). At a high level, the one-way construction treats the input as a secret and produces a single share. The random coins of the function determine which share the function outputs. Intuitively, this construction is extractable because an extractor can query the adversary to get a share, rewind it and query it again until a sufficient number of shares are availabe.

Moreover, it is one-way because of the secrecy of the underlying SS scheme.

Extractable POW functions. We show how to transform any POW function (with additional properties) to extractable POW function and extractable POW functions with auxiliary information (here, auxiliary information is for perfect one-wayness). Informally, our transformation imposes a structure on the new function so that a preimage can be recovered from any two "related" images. Specifically, if $\mathbb{H}$ is the old POW function and x is the input, then an image under the new function, $\mathbb{O}$, consists of some images of $(x, 1)$ and (x, x_i) (x_i is the ith bit of x) under $\mathbb{H}$ for $i = 1, ..., |x|$. Observe that it is easy to recover the i-bit of x from $H_k((x, 1), r)$ and $H_k((x, x_i), r)$. Therefore, an extractor uses rewinding and recovers x by asking the adversary to compute images of $(x, 1)$ and (x, x_i) using the same random coins in two different executions of the game. Depending on the assumptions used, this construction (or a similar one) is perfectly one-way or perfectly one-way with auxiliary information (see Section 4.3.2).

On the relation between extractable functions and Σ-protocols. We remark that a slightly different notion of extractable POW functions can be constructed from any POW function, $\mathbb{H}$, and a Σ-protocol [Blu86, CDN01] for proving preimage knowledge of $\mathbb{H}$. This notion is weaker than the previous one because the construction imposes more restrictions on the adversary. For more information, we refer the reader to Section 4.4.

Noninteractive extraction as interactive extraction. It is worth mentioning that noninteractive extraction can be viewed as a two-round interactive extraction analogous to the three-round extraction discussed above. Specifically, in the first round the challenger sends a random function from the family and the adversary responds with a point in the range of this function. That is, there is a fixed function, g, the challenger sends a random r, and the adversary responds with $g(x, r) = f_r(x)$.

4.1.2 Organization

We give formal definitions of extractable functions in Section 4.2, construct them in Section 4.3, and discuss the connection to Σ-protocols in Section 4.4.

4.2 Definitions

As we discussed in the introduction, interactive extraction of a probabilistic function forces an adversary, A, to compute the function with random coins chosen by an external challenger. This can be rephrased as: A has to be able to compute not only one image but many images of x, e.g., A may be able to compute $H_k(x, r)$ for any r. If A can do so, then x is extractable.

Interactive extraction utilizes a 3-round game between the adversary, A, and a challenger. The latter is a role that an extractor, $\mathcal{K}_A$ may play. The 3-round game (see Figure 4.1) starts with A sending an image, y_0. The challenger sends uniform strings, $r_1, ..., r_n$, and A has to answer with $y_1, ..., y_n$, using $r_1, ..., r_n$ as random coins for $\mathbb{H}$. Preimage extraction means that if there is a common preimage for $y_0, ..., y_n$, then A knows this preimage. As previously discussed, this form of knowledge is captured by the existence of an efficient extractor, $\mathcal{K}_A$, that computes x from the input of A.

Auxiliary information. As we discussed in Section 3.2, we can study extraction in the absence or presence of auxiliary information. Moreover, auxiliary information may be dependent or independent of the specific function under study. Recall that for the case of a single function, the distinction between dependent and independent auxiliary information becomes moot. Moreover, unlike the noninteractive case, where we were not able to realize the definition for a single function, we can do so for interactively-extractable functions. Thus, we are able to realize the strongest notion of interactive extraction: extraction for a single function in the presence of auxiliary information. In fact, we realize an even stronger notion. The extractor is universal in the sense that it is *independent* of the specific adversary (see end of Section 4.2.3 for more detail). However, for completeness, we present the full set of definitions, that is definitions for a single function and for a randomly chosen function, and with and without auxiliary information.

Extraction error. As we discussed in the introduction, unlike the case of noninteractive extraction, we are able to realize interactive extraction only *with arbitrary small but noticeable error*. In other words, for every adversary and every polynomial, p, there is an extractor that fails with probability at most $\frac{1}{p}$. Consequently, the following definitions take into account this noticeable error. Definitions for negligible error can be easily

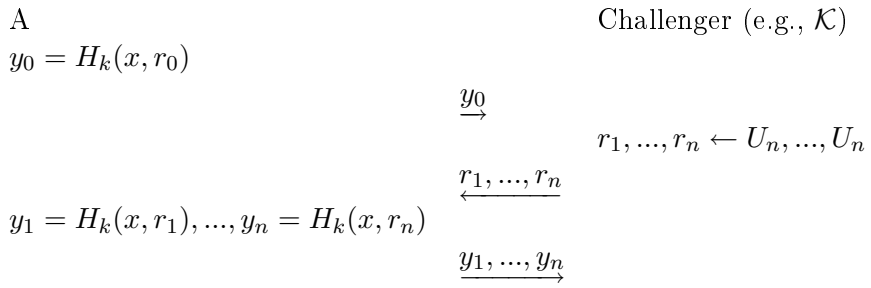

Figure 4.1: The 3-round interaction

obtained from the ones that appear here by removing all occurrences of the polynomial p. We study interactive extraction with negligible error in Chapter 5.

4.2.1 Preimage Knowledge without Auxiliary Information

In the absence of auxiliary information, there are two possible definitions. The first and stronger version requires extraction for any function while the second one applies for a uniformly-chosen function.

Definition 4.2.1 (Interactive extraction without auxiliary information). *A verifiable and probabilistic family ensemble, $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$ is called **interactively extractable** without auxiliary information if for any PPT, A (with private random coins denoted by r_A), and polynomial, p, there exists a PPT, $\mathcal{K}_{A,p}$, such that for any $k \in K_n$:*

$$Pr[(r_1, ..., r_n) \leftarrow R_n, ..., R_n,\ (y_0, s) = A(k, r_A),\ (y_1, ..., y_n) = A(s, r_1, ..., r_n, r_A),$$

$$x \leftarrow \mathcal{K}_{A,p}(k, r_1, ..., r_n, r_A) :$$

$$(V_{\mathbb{H}}(x, y_0) = 1 \text{ and } \forall i \geq 1,\ y_i = H_k(x, r_i)) \text{ or } (\forall x',\ V_{\mathbb{H}}(x', y_0) \neq 1 \text{ or } \exists i,\ y_i \neq H_k(x', r_i))]$$

$$> 1 - \frac{1}{p(n)} - \mu(n).$$

Definition 4.2.2 (Interactive extraction without auxiliary information). *A verifiable and probabilistic family ensemble, $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$ is called **interactively extractable** without auxiliary information if for any PPT, A (with private random coins denoted by r_A), and polynomial, p, there exists a PPT, $\mathcal{K}_{A,p}$, such that:*

$$Pr[k \leftarrow K_n,\ (r_1, ..., r_n) \leftarrow R_n, ..., R_n,\ (y_0, s) = A(k, r_A),\ (y_1, ..., y_n) = A(s, r_1, ..., r_n, r_A),$$

$$x \leftarrow \mathcal{K}_{A,p}(k, r_1, ..., r_n, r_A) :$$

$$(V_{\mathbb{H}}(x, y_0) = 1 \text{ and } \forall i \geq 1,\ y_i = H_k(x, r_i)) \text{ or } (\forall x',\ V_{\mathbb{H}}(x', y_0) \neq 1 \text{ or } \exists i,\ y_i \neq H_k(x', r_i))]$$

$$> 1 - \frac{1}{p(n)} - \mu(n).$$

4.2.2 Preimage Knowledge with Independent Auxiliary Information

As we discussed before, independent auxiliary information applies only when the function is sampled uniformly. In the following definition, $\mathbb{Z}$ refers to any distribution on auxiliary information.

Definition 4.2.3 (Interactive extraction with independent auxiliary information). *A verifiable and probabilistic family ensemble, $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$ is called **interactively extractable** with independent auxiliary information if for any PPT, A (with private random coins denoted by r_A), and polynomial, p, there exists a PPT, $\mathcal{K}_{A,p}$, such that for any distribution $\mathbb{Z} = \{Z_n\}_{n\in\mathbb{N}}$:*

$$Pr[k \leftarrow K_n,\ z \leftarrow Z_n,\ (r_1, ..., r_n) \leftarrow R_n, ..., R_n,\ (y_0, s) = A(k, z, r_A),$$

$$(y_1, ..., y_n) = A(s, r_1, ..., r_n, r_A),\ x \leftarrow \mathcal{K}_{A,p}(k, z, r_1, ..., r_n, r_A) :$$

$$(V_{\mathbb{H}}(x, y_0) = 1 \text{ and } \forall i \geq 1,\ y_i = H_k(x, r_i)) \text{ or } (\forall x',\ V_{\mathbb{H}}(x', y_0) \neq 1 \text{ or } \exists i,\ y_i \neq H_k(x', r_i))]$$

$$> 1 - \frac{1}{p(n)} - \mu(n).$$

4.2.3 Preimage Knowledge with Dependent Auxiliary Information

Recall from Chapter 3 that dependent auxiliary information for noninteractive extraction is restricted to images under the function. However, as we mentioned in the introduction of the current chapter, interactive extraction allows for auxiliary information with arbitrary dependency on the function. The next two definitions capture the notion of interactive extraction with dependent auxiliary information for any function and for a uniformly-chosen function, respectively.

Definition 4.2.4 (Interactive extraction with dependent auxiliary information). *A verifiable and probabilistic family ensemble, $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$ is called **interactively extractable** with dependent auxiliary information if for any PPT, A (with private*

random coins denoted by r_A), and polynomial, p, there exists a PPT, $\mathcal{K}_{A,p}$, such that for any distribution $\mathbb{Z} = \{Z_n\}_{n\in\mathbb{N}}$ and any $k \in K_n$:

$$Pr[z \leftarrow Z_n,\ (r_1, ..., r_n) \leftarrow R_n, ..., R_n,\ (y_0, s) = A(k, z, r_A),$$

$$(y_1, ..., y_n) = A(s, r_1, ..., r_n, r_A),\ x \leftarrow \mathcal{K}_A(k, z, r_1, ..., r_n, r_A) :$$

$$(V_{\mathbb{H}}(x, y_0) = 1 \text{ and } \forall i \geq 1, y_i = H_k(x, r_i)) \text{ or } (\forall x',\ V_{\mathbb{H}}(x', y_0) \neq 1 \text{ or } \exists i,\ y_i \neq H_k(x', r_i))]$$

$$> 1 - \frac{1}{p(n)} - \mu(n).$$

Definition 4.2.5 (Interactive extraction with dependent auxiliary information). *A verifiable and probabilistic family ensemble, $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$ is called **interactively extractable** with dependent auxiliary information if for any PPT, A (with private random coins denoted by r_A), and polynomial, p, there exists a PPT, $\mathcal{K}_{A,p}$, such that for any distribution $\mathbb{Z} = \{Z_n\}_{n\in\mathbb{N}}$ (that may depend on the function):*

$$Pr[k \leftarrow K_n,\ z \leftarrow Z_n(k),\ (r_1, ..., r_n) \leftarrow R_n, ..., R_n,\ (y_0, s) = A(k, z, r_A),$$

$$(y_1, ..., y_n) = A(s, r_1, ..., r_n, r_A),\ x \leftarrow \mathcal{K}_A(k, z, r_1, ..., r_n, r_A) :$$

$$(V_{\mathbb{H}}(x, y_0) = 1 \text{ and } \forall i \geq 1,\ y_i = H_k(x, r_i)) \text{ or } (\forall x',\ V_{\mathbb{H}}(x', y_0) \neq 1 \text{ or } \exists i, y_i \neq H_k(x', r_i))]$$

$$> 1 - \frac{1}{p(n)} - \mu(n).$$

Finally, we point out that our constructions in Section 4.3 allow for a special type of knowledge extraction, namely, universal (blackbox) extraction. That is, there is a universal extractor, $\mathcal{K}$, that can recover a preimage given blackbox access (with rewinding) to any adversary A. Moreover, $\mathcal{K}$ takes a polynomial, p, as input. It runs in time polynomial in p and n, and fails with probability at most $\frac{1}{p} + \mu$. The blackbox version of Definition 4.2.4 follows.

Definition 4.2.6 (Blackbox interactive extraction). *A verifiable and probabilistic family ensemble, $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$ is called **blackbox interactively extractable** (interactively extractable, for short) if there exists a PPT, $\mathcal{K}$, such that for any distribution*

$\mathbb{Z} = \{Z_n\}_{n\in\mathbb{N}}$, *any PPT* A, *any polynomial,* p, *and any* $k \in K_n$*:*

$$Pr[z \leftarrow Z_n,\ (r_1, ..., r_n) \leftarrow R_n, ..., R_n,\ (y_0, s) \leftarrow A(k, z),\ (y_1, ..., y_n) \leftarrow A(s, r_1, ..., r_n),$$

$$x \leftarrow \mathcal{K}^A(k, p) :$$

$$(V_{\mathbb{H}}(x, y_0) = 1 \text{ and } \forall i \geq 1,\ y_i = H_k(x, r_i)) \text{ or } (\forall x',\ V_{\mathbb{H}}(x', y_0) \neq 1 \text{ or } \exists i, y_i \neq H_k(x', r_i))]$$

$$> 1 - \frac{1}{p(n)} - \mu(n).$$

In this chapter, we use Definition 4.2.6 to refer to interactive extraction.

4.3 Constructions

We give one construction of extractable one-way function and two constructions of extractable POW functions. The one-way construction is based on verifiable secret sharing (VSS) schemes [CGMA85, Fel87]. The first extractable POW construction is based on standard perfectly one-way assumptions and achieves both extraction and perfect one-wayness. However, it does not achieve perfect one-wayness *with auxiliary information*. The last construction is based on a stronger perfectly one-way assumption but it achieves perfect one-wayness with auxiliary information. All constructions achieve blackbox extraction with auxiliary information as in Definition 4.2.6.

4.3.1 Extractable One-way Functions

We give a construction of an interactively-extractable one-way function from VSS schemes [CGMA85, Fel87] (with an additional property). For clarity, we start with a construction from SS schemes [Bla79, Sha79] that does not achieve all that we want.

Recall that interactive extraction is relevant to probabilistic functions only. Therefore, this construction is randomized in nature. Informally, a probabilistic one-way function takes two input x and r, where r is the random coins used by the function, and is one-way in x. The one-wayness property is taken over the random choice of r (see Definition 2.2.2).

An initial attempt at constructing extractable one-way functions is to use an SS scheme. Recall from the introduction, that a secret sharing scheme allows a designated

machine, $\mathcal{S}$, to split a secret into n shares such that it is possible to recover the secret if and only if a sufficient number (specifically, u) of shares are present. Formally,

Definition 4.3.1 (Secret sharing). *A pair of polynomial-time machines, $(\mathcal{S}, \mathcal{R})$, is called a (n, t, u)-SS scheme if:*

- ***Correctness:*** *For any secret, s, any n shares $d_1, ..., d_n \leftarrow \mathcal{S}(s)$, and any u subset of the shares $\{a_1, ..., a_n\} \subset \{1, ..., n\}$, $\mathcal{R}((a_1, d_1), ..., (a_u, d_u)) = s$*
- ***Secrecy:*** *For any PPT, A and any t shares that may be chosen adaptively, $a_1, ..., a_t$:*

$$Pr[s \leftarrow U_n \, ; (d_1, ..., d_n) \leftarrow \mathcal{S}(s), \; s' \leftarrow A((a_1, d_1), ..., (a_t, d_t)) : s = s'] \leq \mu(n).$$

Lets examine the following candidate construction from a (n, t, u)-SS scheme, $(\mathcal{S}, \mathcal{R})$. Given an input x, use $\mathcal{S}$ to split x into n shares and output t shares. Formally, the candidate function F receives $(x, r^{\mathcal{S}})$ as input and $a_1, ..., a_t$ as random coins, where $r^{\mathcal{S}}$ denotes the random coins needed to run $\mathcal{S}$, and outputs $(a_1, d_1), ..., (a_t, d_t)$, where $d_1, ..., d_n = \mathcal{S}(x, r^{\mathcal{S}})$. We associate with F the verifier, V_F, which on input $(x, r^{\mathcal{S}})$ and $y = (a_1, d_1), ..., (a_t, d_t)$ accepts if and only if $\mathcal{S}(x, r^{\mathcal{S}}) = d'_1, ..., d'_n$ and $d_1 = d'_{a_1}, ..., d_t = d'_{a_t}$.

The one-wayness of this construction follows directly from the secrecy of the SS-scheme. Moreover, it seems there is a universal extractor that can recover x. In more detail, this extractor, $\mathcal{K}$, has oracle access to the adversary A (with rewinding). $\mathcal{K}$ sends $1, ..., t$ in the second round of the interactive game, to receive $a_1, ..., a_t$, rewinds A back to step 2, sends $t+1, ..., 2t$ to get $a_{t+1}, ..., a_{2t}$ and so on until $\mathcal{K}$ has u shares and then it simulates $\mathcal{R}$ to recover x. However, this reasoning is not entirely correct! The problem with this construction is that the first message in the game is not binding. In other words, there may be two executions of the game with the same first message in both of them but the preimages of these two executions are not the same. For instance, suppose we have a $(n, t, t+1)$-SS scheme. Then, A may send $d_1, ..., d_t$ in one run of the game and $d_{t+1}, ..., d_{2t}$ in another but $d_1, ..., d_{t+1}$ and $d_2, ..., d_{t+2}$ may yield two distinct preimages. Even worse, it is conceivable that A sends $d_1, ..., d_t$ in one run and $d_{t+1}, ..., d_{2t}$ in the second run but there is no common preimage for any $t+1$ elements from $d_1, ..., d_{2t}$. This binding problem can be solved using VSS schemes which guarantee commitment through

verification of share validity.

Recall that a VSS scheme is an SS scheme that permits checking the validity of each share and ensures uniqueness of the secret that $\mathcal{R}$ recovers on any u shares. Formally,

Definition 4.3.2 (Verifiable secret sharing). *A triple of polynomial-time machines, $(\mathcal{S}, \mathcal{R}, \mathcal{V})$, is called a (n, t, u)-VSS scheme if:*

- ***SS:*** *$(\mathcal{S}, \mathcal{R})$ is (n, t, u)-SS scheme as in Definition 4.3.1.*
- ***Verification correctness:*** *For any secret, s, any n shares $d_1, ..., d_n \leftarrow \mathcal{S}(s)$, and any index i, $\mathcal{V}(i, d_i) = 1$.*
- ***Verification soundness:*** *For any i, d_i where $\mathcal{V}(i, d_i) = 1$, there exists an x such that for any $(a_1, d_{a_1}), ..., (a_{u-1}, d_{a_{u-1}})$, where $\mathcal{V}(a_i, d_{a_i}) = 1$ $\forall 1 \leq i \leq u-1$, $\mathcal{R}((i, d_i), (a_1, d_{a_1}), ..., (a_{u-1}, d_{a_{u-1}})) = x$.*

The actual construction is more efficient than the one described above in that it outputs a single share. Formally,

Construction 4.3.1. *Let $(\mathcal{S}, \mathcal{R}, \mathcal{V})$ be a VSS scheme. Then, let $\mathbb{F} = \{F_n\}_{n \in \mathbb{N}}$ be the following randomized family ensemble:*

$$F_n(x = (s, r^{\mathcal{S}}), i) = i, d_i,$$

where $i \in \{1, ..., n\}$ and $d_1, ..., d_n = \mathcal{S}(s, r^{\mathcal{S}})$.

We associate with Construction 4.3.1, the verifier, $V_{\mathbb{F}}$, where $V_{\mathbb{F}}(x, y) = 1$ if and only if $\mathcal{V}(y) = 1$ and x can be written as $s, r^{\mathcal{S}}$ and y as i, d_i and $\mathcal{S}(s, r^{\mathcal{S}}) = d'_1, ..., d'_n$ and $d'_i = d_i$.

Another issue with this construction is that Definition 4.3.1 implies that an extractor can recover x but may not recover $r^{\mathcal{S}}$, which is part of the input to F. To recover $r^{\mathcal{S}}$, we assume that $\mathcal{R}$ does so on u shares. Note that this is true for Shamir secret-sharing [Sha79], where $r^{\mathcal{S}}$ constitutes the coefficients (minus the zero term) for the polynomial used in sharing the secret. Formally, we assume the following strong version of correctness:

Definition 4.3.3 (Strong correctness). *A (n, t, u)-SS scheme, $(\mathcal{S}, \mathcal{R})$, is **strongly correct** if for any secret, s, any n shares $d_1, ..., d_n = \mathcal{S}(s, r^{\mathcal{S}})$, and any u subset of the shares $\{a_1, ..., a_n\} \subset \{1, ..., n\}$, $\mathcal{R}((a_1, d_1), ..., (a_u, d_u)) = s, r^{\mathcal{S}}$.*[1]

We show that this construction is an extractable one-way function based on the assumption that $(\mathcal{S}, \mathcal{R}, \mathcal{V})$ is a $(n^2, n+1, n+2)$-VSS scheme with strong correctness.

Theorem 4.3.1. *Let $(\mathcal{S}, \mathcal{R}, \mathcal{V})$ be a strongly-correct $(n^2, n+1, n+2)$-VSS scheme (as in Definitions 4.3.2 and 4.3.3) then Construction 4.3.1 is an extractable one-way function (as in Definitions 2.2.2 and 4.2.6).*

Proof. **One-wayness.** One-wayness follows directly from secrecy of the SS scheme and uniqueness of the secret. Moreover, by the same definition, this function remains one-way after one run of the interactive game.

Extraction. This is a proof by construction. We present a black-box extractor that works with probability polynomially close to 1. In more detail, we present a PPT extractor having black-box access (with rewinding) to any PPT machine that plays the interactive game of Definition 4.2.6. This extractor also receives a polynomial bound, which represents the allowed margin of error, and halts in time polynomially related to this bound. Let $k = (\mathcal{S}, \mathcal{R}, \mathcal{V})$, then $\mathcal{K}$ receives also k. Wlog, assume that $r^{\mathcal{S}}$ has domain $\{0,1\}^n$.

Formally, the extractor, $\mathcal{K}$, works as defined in Algorithm 4.3.1.

Analysis

The rest of the proof shows that $\mathcal{K}$ satisfies Definition 4.2.6. Informally, we show that for any PPT, A, if for some input (z, r_A), A succeeds with some inverse polynomial probability, say $\frac{1}{p}$, in answering the challenges, then $\mathcal{K}^A(k, p)$ almost always extracts a preimage. In other words, this extractor fails in extracting a preimage only on input that causes A to succeed with probability *less* than $\frac{1}{p}$. Thus, its failure probability is at most $\frac{1}{p}$.

In more detail, we prove our claim by showing the existence of a big set of random challenges that A can answer consistently. The extractor then needs to sample uniform challenges multiple times in order to ensure sampling from this favorable set. Once it samples $n+1$ elements from this set, it can use $\mathcal{R}$ to extract a point x. We then use

[1] Strong correctness changes verification soundness of VSS schemes to force a unique $x, r^{\mathcal{S}}$ (instead of x) for any valid pair i, d_i.

input : $(\mathcal{S}, \mathcal{R}, \mathcal{V}), p$
interaction: with an external PPT, A

receive $y_0 = (a_0, d_{a_0})$;
$B = \{a_0\}$;
$C = \{y_0\}$;
for $m = 1$ **to** $n + 1$ **do**
    for $j = 1$ **to** n **do**
        $b \leftarrow \{1, ..., n^2\} \setminus B$;
        for $i = 1$ **to** $2n^2 p(n)$ **do**
            $(a_1, ..., a_n) \leftarrow \{1, ..., n^2\}$;
            $l \leftarrow \{0, ..., n - 1\}$;
            send $a_1, ..., a_l, b, a_{l+2}, ..., a_n$;
            receive $y_1, ..., y_n$;
            rewind A;
            if $y_{l+1} = (b, y')$ **and** $\mathcal{V}(y_{l+1}) = 1$ **then**
                $B = B \cup \{b\}$;
                $C = C \cup \{(b, y_{l+1})\}$;
                break *innermost two loops*;
        end
    end
end
if $|C| = n + 2$ **then**
    return $\mathcal{R}(C)$;
$s, r^{\mathcal{S}} \leftarrow U_n$;
return $s, r^{\mathcal{S}}$;

Algorithm 4.3.1: $\mathcal{K}$

verification soundness of the VSS scheme to argue that x is a preimage of y_0. Using verification soundness again, we show that x is also a preimage of $y_1, ..., y_n$.

Formally, denote by $R^{\mathbb{F}}$ the domain of random coins for $\mathbb{F}$. Suppose that for some k, z, r_A and some polynomial, p, we have:

$$Pr[(r_1^F, ..., r_n^F) \leftarrow (R_n^F, ..., R_n^F),\ A(k, z, r_A) = (y_0, s),\ A(s, r_1^F, ..., r_n^F) = (y_1, ..., y_n) :$$

$$\exists x',\ V_{\mathbb{F}}(x', y_0) = 1 \text{ and } \forall i \geq 1,\ y_i = F_k(x', r_i^F)] \geq \frac{1}{p(n)}. \tag{4.1}$$

Then, we show that for the same k, z, r_A:

$$Pr[(r_1^F, ..., r_n^F) \leftarrow (R_n^F, ..., R_n^F),\ A(k, z, r_A) = (y_0, s),\ A(s, r_1^F, ..., r_n^F) = (y_1, ..., y_n),$$

$$x \leftarrow \mathcal{K}^A(k, p) : V_{\mathbb{F}}(x, y_0) = 1] > 1 - \mu(n). \tag{4.2}$$

Recall that each r_i^F is taken from $R^{\mathbb{F}} = \{1, ..., n^2\}$. Eq. 4.1 can be rephrased as: there exists a subset $S \subseteq \underbrace{(R_n^F, ..., R_n^F)}_{n}$, $|S| \geq \frac{n^{2n}}{p(n)}$ such that for all $(r_1^F, ..., r_n^F) \in S$, $A(k, z, r_A) = (y_0, s)$, $A(s, r_1^F, ..., r_n^F) = (y_1, ..., y_n)$, and $\exists x',\ \forall i,\ y_i = F_k(x', r_i^F)$.

Now, recall that each $r_i^F = j$. For each such j, let t_j denote the number of times that j appears in any vector in S. Furthermore, let S' be a new set that contains all the vectors in S except those that contain a j with $t_j \leq \frac{n^{2n}}{n^2 2p(n)}$. Since for each j at most $\frac{n^{2n}}{n^2 2p(n)}$ vectors are deleted from S and there are at most n^2 such j, $|S'| \geq \frac{n^{2n}}{2p(n)}$. Now, let T be the set of j that appear in any vector in S'. We should have:

$$|T|^n \geq |S'| \geq \frac{n^{2n}}{2p(n)}$$

$$\implies |T| \geq \frac{n^2}{(2p(n))^{\frac{1}{n}}} \geq \frac{n^2}{2},$$

where the last inequality holds for sufficiently large n.

Therefore, the probability that $\mathcal{K}$ does not find, in line 6 of its code and for all n repetitions, some b that belongs to T is less than $\frac{1}{2^n}$. Now, suppose that b happens to be in T. The probability that a uniformly chosen vector $r = r_1^F, ..., r_n^F$, conditioned on

containing b, falls in S is:

$$Pr[(r_1^F, ..., r_n^F) \leftarrow r_n^F \times ... \times r_n^F : (r_1^F, ..., r_n^F) \in S | \exists i : r_i^F = b]$$

$$\geq \frac{\frac{n^{2n}}{n^2 2p(n)}}{\frac{n^{2n+1}}{n^2}} = \frac{1}{2np(n)}$$

So that, if $\mathcal{K}$ finds, in line 6 of the code and in some iteration, some b that belongs to T, the probability that vector $r_1^F, ..., r_n^F$ sent to A in line 10 is in S is $\frac{1}{2np(n)}$. Repeated sampling for $2n^2p(n)$ times ensures that the probability of failure in all $2n^2p(n)$ iterations is negligible. Thus, for any iteration of the outermost loop, the probability of failing to find a vector $r_1^F, ..., r_n^F$ in S remains negligible.

Now, if $r_1^F, ..., r_n^F \in S$ then by definition of S, $\mathcal{V}(b, y_l) = 1$ (l is the position of b in this vector, see line 9). Consequently, $b, y_l \in C$. So, after the outermost loop ends, T contains $n + 2$ vectors that are valid under $\mathcal{V}$. Let $s, r^{\mathcal{S}} = \mathcal{R}(T)$. We argue that $s, r^{\mathcal{S}}$ is a valid preimage of y_0. Wlog, let $y_0 = 1, y_0'$. Since y_0 is a valid preimage under F, there exists $s', u^{\mathcal{S}}$ such that $V_F((s', u^{\mathcal{S}}), y_0) = 1$. Suppose for the purpose of contradiction that $s, r^{\mathcal{S}} \neq s', u^{\mathcal{S}}$. Compute $d_1, ..., d_{n^2} = \mathcal{S}(s, r^{\mathcal{S}})$ and $d_1', ..., d_{n^2}' = \mathcal{S}(s', u^{\mathcal{S}})$. By definition of verifiable secret sharing, we have $\mathcal{V}(1, d_1) = ... = \mathcal{V}(n^2, d_{n^2}) = \mathcal{V}(1, d_1') = ... = \mathcal{V}(n^2, d_{n^2}') = 1$. Moreover, by strong correctness of secret sharing, $\mathcal{R}(y_0, d_2', ..., d_{n+2}') = s', u^{\mathcal{S}}$. Recall that $s, r^{\mathcal{S}} = \mathcal{R}(T)$ and $y_0 \in T$. This contradicts verification soundness. Consequently, $s, r^{\mathcal{S}} = s', u^{\mathcal{S}}$ and $s, r^{\mathcal{S}}$ is a preimage of y_0. This proves Eq. 4.2.

To prove that $\mathcal{K}$ satisfies Definition 4.2.6, we utilize verification soundness again to show that $s, r^{\mathcal{S}}$ is a preimage of $y_1, .., y_n$. We know that $\mathcal{K}$ computes a preimage, $s, r^{\mathcal{S}}$, of y_0. Moreover, if A can compute some other y_i for which $s, r^{\mathcal{S}}$ is not a preimage but there is another common preimage for y_0 and y_i, then this violates verification soundness. Formally, for any k, z, r_A:

$$Pr[(r_1^F, ..., r_n^F) \leftarrow (r_n^F, ..., r_n^F),\ (y_0, s) = A(k, z, r_A),\ (y_1, ..., y_n) = A(s, r_1^F, ..., r_n^F),$$

$$x \leftarrow \mathcal{K}^A(k, p) :$$

$$V_F(x, y_0) = 1 \text{ and } (\exists i,\ y_i \neq F_k(x, r_i^F)) \text{ and } (\exists x',\ V_F(x', y_0) = 1 \text{ and } \forall i,\ y_i = F_k(x', r_i^F))]$$

$$\leq \mu(n) \tag{4.3}$$

Suppose, for the purpose of contradiction, Eq. 4.3 is not true. Let $d_1, ..., d_{n^2} = \mathcal{S}(x)$ and $d'_1, ..., d'_{n^2} = \mathcal{S}(x')$. Let $D \subset \{d_1, ..., d_{n^2}\} \setminus \{y_0\}$ and $D' \subset \{d'_1, ..., d'_{n^2}\} \setminus \{y_0\}$. By strong correctness, we have for any $d_1, ..., d_{n+2} \in D$, $\mathcal{R}(d_1, ..., d_{n+2}) = x$ and for any $d'_1, ..., d'_{n+2} \in D'$, $\mathcal{R}(d'_1, ..., d'_{n+2}) = x'$. Since $\mathcal{V}(y_0) = 1$ then by verification soundness $\mathcal{R}(y_0, d_1, ..., d_{n+1}) = x$ and $\mathcal{R}(y_0, d'_1, ..., d'_{n+1}) = x'$. However, this contradicts verification soundness unless $x = x'$. Combining Eq. 4.1, 4.2, and Eq. 4.3 finishes the proof.

□

4.3.2 Extractable POW Functions

We give two constructions of extractable POW functions. The first one is perfectly one-way while the second one is perfectly one-way *with auxiliary information*.

The idea behind both constructions is to have pairs of related images with the property that it is easy to compute a preimage if both of them are available. In more detail, we define for every r, an $\hat{r}$, such that $O(x, r), O(x, \hat{r})$ reveals x. So, the extractor can recover a preimage by sending r in the second round of the game in Definition 4.2.6, to get $O(x, r)$, rewinding A, and then sending $\hat{r}$ in the second round of the game to get $O(x, \hat{r})$. On the other hand, for $r_1, ..., r_n$ chosen uniformly and independently, $O(x, r_1), ..., O(x, r_n)$ do not reveal x because it is unlikely that some r_i, r_j satisfy the relation $r_j = \hat{r_i}$. We go into more details after we present the first construction.

4.3.2.1 Extractable POW Functions without Auxiliary Information

Construction 4.3.2. *Let $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$ and $\mathbb{G} = \{\mathbf{G}^n\}_{n\in\mathbb{N}}$ be two verifiable family ensembles, where $H_k : \{0,1\}^n \times R_n \to \{0,1\}^{l(n)}$ and $G_k : \{0,1\}^n \times R_n \to \{0,1\}^{l(n)}$ for some polynomial l. Denote by $\mathbb{O} = \{\mathbf{O}^n\}_{n\in\mathbb{N}}$, where $O_k : \{0,1\}^n \times R_n^O = (R_n^{(4)} \times R_{l(n)}^{(n)}) \to \{0,1\}^{2|r_0^1|+2l(n)+nl(l(n))}$, the family ensemble defined as:*

$$O_k(x, (r_0^1, r_0^2, r_0^3, r_1..., r_n, r_G)) = r_0^2, r_0^3, H_k(x, r_0^1), H_k(t_1, r_1), ..., H_k(t_n, r_n), G_k(x, r_G),$$

where for all i, $t_i = H_k(x, r_0^2)$ if $x_i = 1$, and $t_i = H(x, r_0^3)$ otherwise.

4.3.2.1.1 Extraction. For simplicity, and to see why Construction 4.3.2 is extractable assume that A receives only a single challenge (instead of n), r^O, in the second round of the extraction game. Informally, $\mathcal{K}$ tries to make A output two "related" images that allow it to recover x. To this end, $\mathcal{K}$ sends r^O as a challenge to A, rewinds A, and then sends $\hat{r}^O$. So, if both interactions are consistent, x can be recovered. In more detail, $\mathcal{K}$ sends $\boxed{r_0^1}, r_0^2, r_0^3, r_1, ..., r_n$ to A in the first interaction, where all strings are uniform. In the second interaction, $\mathcal{K}$ sends $u_0^1, \boxed{r_0^1}, u_0^3, u_1, ..., u_n$, where $u_0^1, u_0^3, u_1, ..., u_n$ are chosen uniformly but r_0^1 appears in the box in the first interaction. If A answers both challenges consistently, then $\mathcal{K}$ can recover x. This is so because the third-round message of the first interaction contains $t = H_k(x, r_0^1)$, while the third-round message of the second interaction contains $H_k(t, u_i)$ if and only if the ith bit of x is 1.

We remark that the technical proof requires $\mathbb{H}$ to satisfy a strong form of collision resistance. Specifically, we assume that it is hard to compute three images, y_0, y_1, y_2, the last two using randomness sampled by the challenger, such that there is a common preimage for y_0 and y_1, another common preimage for y_0, y_2 but no common preimage for all three images. It can be shown that strong collision resistance implies collision resistance. Moreover, we show in Section 4.3.2.3 that collision resistance (and a strong perfect one-way assumption) implies strong collision resistance (and injection). The formal definition follows.

Definition 4.3.4 (Strong Collision Resistance). *A verifiable family ensemble, $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$, where for some polynomial $l(n)$, for any $n \in \mathbb{N}$, and any $k \in K_n$, $H_k : \{0,1\}^n \times R_n \to \{0,1\}^{l(n)}$, satisfies **strong collision resistance** if for any PPT A, and $k \in K_n$:*

$$Pr[(r_1, r_2) \leftarrow (R_n, R_n),\ (y_0, s) \leftarrow A(k),\ (y_1, y_2) \leftarrow A(s, r_1, r_2)\ :$$

$$\exists x_1, x_2, x_1 \neq x_2, V_{\mathbb{H}}(x_1, y_0) = V_{\mathbb{H}}(x_2, y_0) = 1$$

$$and\ y_1 = H_k(x_1, r_1)\ and\ y_2 = H_k(x_2, r_2)\ and\ y_2 \neq H_k(x_1, r_2)] \leq \mu(n)$$

Before we present the extraction theorem, we specify the verifier for $\mathbb{O}$, denoted by $V_{\mathbb{O}}$. $V_{\mathbb{O}}(x, y = (r_1, r_2, y_0, ..., y_{n+1})) = 1$ if and only if $V_{\mathbb{G}}(x, y_{n+1}) = 1$ and $V_{\mathbb{H}}(x, y_0) = 1$ and $V_{\mathbb{H}}(t_i, y_i) = 1$ for all $1 \leq i \leq n$, where $t_i = H(x, r_{x_{i+1}})$ and x_i is the i^{th} bit of x.

Theorem 4.3.2. *If $\mathbb{H}$ and $\mathbb{G}$ are two verifiable (as in Definition 2.5.1) family ensembles with public randomness and* ***one of them*** *is strongly collision resistant (as in Definition 4.3.4), then $\mathbb{O}$, the result of applying Construction 4.3.2 on $\mathbb{H}$ and $\mathbb{G}$, is extractable with auxiliary information (as in Definition 4.2.6).*

Proof. This is a proof by construction and is very similar to the proof of Theorem 4.3.1. We present a black-box extractor that works with probability polynomially close to 1. In more detail, we present a PPT extractor having black-box access (with rewinding) to any efficient adversary that plays the interactive game of Definition 4.2.6. This extractor also receives a polynomial bound, which represents the allowed margin of error, and halts in time polynomially related to this bound.

The extractor, $\mathcal{K}$, is formally defined in Algorithm 4.3.2.

input : k, p
interaction: with an external PPT, A

receive y_0;
for $j = 1$ **to** n **do**
  $r_0^1 \leftarrow R_n$;
  for $i = 1$ **to** $16n^3p^2(n)$ **do**
    $(r_0^2, r_0^3, r_1, ..., r_n, r_G) \leftarrow R_n, ..., R_n$;
    $(u_0^1, u_0^3, u_1, ..., u_n, u_G) \leftarrow R_n, ..., R_n$;
    $r_1^O = (r_0^1, r_0^2, r_0^3, r_1, ..., r_n, r_G)$;
    $u_1^O = (u_0^1, r_0^1, u_0^3, u_1, ..., u_n, u_G)$;
    $r_2^O, ..., r_n^O \leftarrow R_n^O, ..., R_n^O$;
    $u_2^O, ..., u_n^O \leftarrow R_n^O, ..., R_n^O$;
    $d_1, d_2 \leftarrow \{1, ..., n\}, \{1, ..., n\}$;
    send $r_2^O, ..., r_{d_1}^O, r_1^O, r_{d_1+1}^O, ..., r_n^O$;
    receive $y_1, ..., y_n$;
    rewind A;
    send $u_2^O, ..., u_{d_2}^O, u_1^O, u_{d_2+1}^O, ..., u_n^O$;
    receive $v_1, ..., v_n$;
    rewind A;
    parse y_{d_1} **as** $r_0^2, r_0^3, y_{d_1}^0, y_{d_1}^1, ..., y_{d_1}^n, y_{d_1}^G$;
    $\alpha = y_{d_1}^0$;
    parse v_{d_2} **as** $r_0^1, u_0^3, v_{d_2}^0, v_{d_2}^1, ..., v_{d_2}^n, v_{d_2}^G$;
    $x = V_{\mathbb{H}}(\alpha, v_{d_2}^1), ..., V_{\mathbb{H}}(\alpha, v_{d_2}^n)$;
    if $V_{\mathbb{O}}(x, y_0) = 1$ **and** $\forall i \geq 1,\ y_i = O_k(x, r_i^O)$ **and** $\forall i \geq 1,\ v_i = O_k(x, u_i^O)$ **then**
      return x;
  end
end
$x \leftarrow U_n$;
return x;

Algorithm 4.3.2: $\mathcal{K}$

Analysis

The rest of the proof shows that $\mathcal{K}$ satisfies Definition 4.2.6. Informally, we show that if for some input (z, r_A), A succeeds with some inverse polynomial probability, say $\frac{1}{p}$, in answering the challenges, then $\mathcal{K}^A(k,p)$ almost always extracts a preimage. In other words, this extractor fails in extracting a preimage only on an input that causes A to succeed with probability *less* than $\frac{1}{p}$. Thus, its failure probability is at most $\frac{1}{p}$.

In more detail, we prove our claim by showing the existence of a big set of random challenges that A can answer consistently. The extractor then needs to sample uniform challenges multiple times in order to ensure sampling from this favorable set. Once it samples two related elements from this set, it can use them to extract a preimage of y_0. Finally, we use strong collision resistance to conclude that such a preimage is also a preimage of $y_1, ..., y_n$.

Formally, suppose that for some k, z, r_A and some polynomial, p, we have:

$$Pr[(r_1^O, ..., r_n^O) \leftarrow (R_n^O, ..., R_n^O),\ A(k, z, r_A) = (r_0^O, y_0, s),\ A(s, r_1^O, ..., r_n^O) = (y_1, ..., y_n) :$$

$$\exists x',\ \forall i,\ y_i = O_k(x', r_i^O)] \geq \frac{1}{p(n)}.^{2} \tag{4.4}$$

Then, we show that for the same k, z, r_A:

$$Pr[A(k, z, r_A) = (r_0^O, y_0, s),\ x \leftarrow \mathcal{K}^A(k,p) : y_0 = O_k(x, r_0^O)] > 1 - \mu(n). \tag{4.5}$$

Recall that each r_i^O is taken from $R_n^O = (R_n^{(4)} \times R_{l(n)}^{(n)})$. Without loss of generality, assume that $R_n = \{0,1\}^n$ and $R_{l(n)} = \{0,1\}^{l(n)}$.

Eq. 4.4 can be rephrased as: there exists a subset $S \subseteq \underbrace{(R_n^O, ..., R_n^O)}_{n}$, $|S| \geq \frac{2^{n^2(l(n)+4)}}{p(n)}$ such that for all $(r_1^O, ..., r_n^O) \in S$,$A(k, z, r_A) = (r_0^O, y_0, s)$, $A(s, r_1^O, ..., r_n^O) = (y_1, ..., y_n)$, and $\exists x',\ \forall i,\ y_i = O_k(x', r_i^O)$.

Now, recall that each $r_i^O = r_0^1, r_0^2, r_0^3, r_1, ..., r_n, r_G$. So, for each $r \in \{0,1\}^n$, let t_r^1 (respectively, t_r^2) denote the number of times that r appears as r_0^1 (respectively, r_0^2) in any r_i^O in any vector in S. Furthermore, let S' be a new set that contains all the vectors in S except those that contain an r as r_0^1 with $t_r^1 \leq \frac{2^{n^2(l(n)+4)}}{2^n 4p(n)}$ or r as r_0^2 with $t_r^2 \leq \frac{2^{n^2(l(n)+4)}}{2^n 4p(n)}$.

[2] r_0^O is in the clear because $\mathbb{H}$ and $\mathbb{G}$ (and consequently, $\mathbb{O}$) have public randomness.

Since for each r at most $\frac{2^{n^2(l(n)+4)}}{2^n 2p(n)}$ vectors are deleted from S and there are at most 2^n such r, $|S'| \geq \frac{2^{n^2(l(n)+4)}}{2p(n)}$. Now, let T_1 (respectively, T_2) be the set of elements that occur as r_0^1 (respectively, r_0^2) in any r_i^O in any vector in S'. Let $T = T_1 \cap T_2$. We should have:

$$|T_1|^n |T_2|^n 2^{n^2(l(n)+2)} \geq |S'| \geq \frac{2^{n^2(l(n)+4)}}{2p(n)}$$

$$\implies |T_1||T_2| \geq \frac{2^{2n}}{(2p(n))^{\frac{1}{n}}} \geq 2^{2n-\frac{clogn}{n}}$$

$$\implies |T_1| \geq 2^{n-\frac{clogn}{n}} \; and \; |T_2| \geq 2^{n-\frac{clogn}{n}}$$

$$\implies |T| > \frac{2^n}{2},$$

where c is some constant that depends on p, and the last inequality holds for sufficiently large n.

Therefore, the probability that $\mathcal{K}$ does not find, in line 3 of its code and for all n repetitions, r_0^1 that belongs to T is less than $\frac{1}{2^n}$. Now, suppose that r_0^1 happens to be in T. The probability that a uniformly chosen vector $r = r_1^O, ..., r_n^O$, conditioned on containing r_0^1 as a first entry in some r_i^O, falls in S is:

$$Pr[(r_1^O, ..., r_n^O) \leftarrow R_n^O \times ... \times R_n^O : (r_1^O, ..., r_n^O) \in S | \exists i : r_i^O = (r_0^1, r_0^2, r_0^3, r_1, ..., r_n, r_G)]$$

$$\geq \frac{\frac{2^{n^2(l(n)+4)}}{2^n 4p(n)}}{\frac{n 2^{n^2(l(n)+4)}}{2^n}} = \frac{1}{4np(n)}$$

The same inequality holds when considering r_0^1 as the second entry in some r_i^O.

So that, if $\mathcal{K}$ finds, in line 3 of the code and in some iteration, an r_0^1 that belongs to T, the probability that both vectors $r_1^O, ..., r_n^O$ and $u_1^O, ..., u_n^O$, computed in lines $5-10$, are in S is $\frac{1}{(4np(n))^2}$. Repeated sampling for $16n^3p^2(n)$ times ensures that the probability of failure in all $16n^3p^2(n)$ iterations is negligible.

Now, if both vectors $r_1^O, ..., r_n^O$ and $u_1^O, ..., u_n^O$ are in S, then by definition of S, A will compute consistent $y_1, ..., y_n$ and $v_1, ..., v_n$ in lines $13-16$. In other words, $\exists x_1,\ V_{\mathbb{O}}(x_1, y_0) = 1,$ and $\forall i,\ y_i = O_k(x_1, r_i^O)$, and $\exists x_2,\ V_{\mathbb{O}}(x_2, y_0) = 1,$ and $\forall i,\ v_i = O_k(x_2, u_i^O)$. However, by strong collision resistance ($\mathbb{O}$ is strongly collision resistance because either $\mathbb{H}$ or $\mathbb{G}$ is), there is a common preimage for $y_1, ..., y_n$ and $v_1, ..., v_n$. Formally, $\exists x',\ V_{\mathbb{O}}(x', y_0) = 1,$ and $\forall i,\ y_i = O_k(x', r_i^O)$ and $v_i = O_k(x', u_i^O)$. Therefore, α as com-

puted in line 19 is equal to $H_k(x', r_0^1)$. Moreover, $v_{d_2} = O_k(x', (u_0^1, r_0^1, u_0^3, u_1, ..., u_n, u_G))$. Now, we need to show that x as computed in line 21 is equal to x'. Observe that if the ith bit of x' is 1 then $x_i = V_{\mathbb{H}}(\alpha, v_{d_2}^i) = 1$. On other hand, if the ith bit of x' is 0, we show that the corresponding bit of x is also 0. We know that if $x'_i = 0$ then $V_{\mathbb{H}}(H_k(x', u_0^3), v_{d_2}^i) = 1$. Since r_0^1 and u_0^3 are chosen uniformly and $\mathbb{H}$ has public randomness, the probability that $\alpha = H_k(x', r_0^1)$ is equal to $H_k(x', u_0^3)$ is negligible. Therefore, by collision resistance, $V_{\mathbb{H}}(\alpha, v_{d_2}^i)$ and hence x_i are almost always equal to 0. Line 22 verifies that x is a valid preimage.

This proves Eq. 4.5. To prove that $\mathcal{K}$ satisfies Definition 4.2.6, we utilize strong collision resistance. We know that $\mathcal{K}$ can compute a preimage, x, of y_0. Moreover, if A can compute some other y_i for which x is not a preimage but there is another common preimage for y_0 and y_i, then this violates strong collision resistance. Formally,

$$Pr[z \leftarrow Z_n,\ (r_1^O, ..., r_n^O) \leftarrow (R_n^O, ..., R_n^O),\ (r_0^O, y_0, s) \leftarrow A(k, z),$$

$$(y_1, ..., y_n) \leftarrow A(s, r_1^O, ..., r_n^O), x \leftarrow \mathcal{K}^A(k, p) :$$

$$y_0 = O_k(x, r_0^O) \text{ and } (\exists i,\ y_i \neq O_k(x, r_i^O)) \text{ and } (\exists x',\ \forall i,\ y_i = O_k(x', r_i^O))] \leq \mu(n) \quad (4.6)$$

Suppose, for the purpose of contradiction, Eq. 4.6 does not hold. Then, there is a PPT, B, that violates strong collision resistance. B first runs A to compute y_0 in the first phase. Note that y_0 has two distinct preimages, x as computed by $\mathcal{K}$ and x', the common preimage of $y_0, ..., y_n$. In the second phase, B receives as input two random challenges. It computes, on its own, an image, y_1, of x under the first random challenge (it uses $\mathcal{K}$ to find x), and asks A to compute an image, y_2, of x' under the second challenge. By the negation of Eq. 4.6, with a nonnegligible probability, y_2 is not a valid image of x, contradicting strong collision resistance.

Combining Eq. 4.4, 4.5, and Eq. 4.6 finishes the proof.

□

Perfect one-wayness. Construction 4.3.2 uses two functions, $\mathbb{H}$ and $\mathbb{G}$, instead of one due to the properties needed to prove perfect one-wayness and extraction. Specifically, our proof of perfect one-wayness uses the assumption that the function, $\mathbb{H}$, is *statistically* perfectly one-way. On the other hand, extraction uses a strong collision resistance as-

sumption on the underlying function. Currently, we know of only one class of functions satisfying strong collision resistance, namely statistically binding functions. However, no single function can be both statistically hiding and statistically binding. Therefore, we use two functions. We assume that $\mathbb{G}$ (and consequently $\mathbb{O}$) is strongly collision resistant, e.g., statistically binding, so that $\mathbb{O}$ is extractable. On the other hand, $\mathbb{H}$ is assumed to be a statistical POW function. So, if we exclude the image under $\mathbb{G}$ from the output of $\mathbb{O}$, we get a statistical POW function. Therefore, if $\mathbb{G}$ is computationally perfectly one-way with auxiliary information (it is sufficient for the auxiliary information to be only a statistically hiding function), then $\mathbb{O}$ is a *computational* POW function.

Theorem 4.3.3. *Suppose $\mathbb{H}$ satisfies statistical $(2n+1)t$-pseudorandomness (as in Definition 2.5.4) and has public randomness and $\mathbb{G}$ satisfies computational t-pseudorandomness with respect to auxiliary information (as in Definition 2.5.6), then Construction 4.3.2 satisfies computational t-pseudorandomness (as in Definition 2.5.6).*

Proof. For simplicity, we prove the special case of 1-pseudorandomness (when $\mathbb{H}$ satisfies statistical $(2n+1)$-pseudorandomness and $\mathbb{G}$ satisfies computational 1-pseudorandomness with auxiliary information). The general case of t-pseudorandomness is similar.

The proof consists of two steps. First, we prove that the first part of $\mathbb{O}$, specifically,

$$O'_k(x,(r_0^1,r_0^2,r_0^3,r_1...,r_n)) \triangleq r_0^2,r_0^3,H_k(x,r_0^1),H_k(t_1,r_1),...,H_k(t_n,r_n)$$

is statistically close to uniform. Then, we combine this claim with the fact that $\mathbb{G}$ is computational pseudorandom with auxiliary information to conclude that $\mathbb{O}$ is computationally perfectly one-way.

$O'_k(x,.)$ is Statistically Close to Uniform. Let $O_k^0(x,r^O=(r_0^1,r_0^2,r_0^3,r_1,...,r_{2n})) \triangleq$

$$r_0^2,r_0^3,H_k(x,r_0^1),H_k(H_k(x,r_0^2),r_1),...,H_k(H_k(x,r_0^2),r_n),$$

$$H_k(H_k(x,r_0^3),r_2),...,H_k(H_k(x,r_0^3),r_{2n}).$$

and

$$O_k^n(x,r^O=(r_0^1,r_0^2,r_0^3,r_1,...,r_{2n})) \triangleq r_0^2,r_0^3,H_k(x,r_0^1),H_k(t_1,r_1),...,H_k(t_{2n},r_{2n})),$$

where $t_i = H_k(x, r_0^2)$ and $t_{n+i} = H_k(x, r_0^3)$ if $x_i = 1$ and vice versa if $x_i = 0$.

Note that $\mathbb{O}'$ is a substring (in particular, the first $n+3$ strings) of O^n.

We have for any well-spread distribution, $\mathbb{X}$:

$$\Delta((H_k(X_n, R_n^1), H_k(X_n, R_n^2), H_k(X_n, R_n^3)), (U_n^1, U_n^2, U_n^3)) < \mu(n) \tag{4.7}$$

$$\implies \Delta(O_k^0(X_n, R_n^{O^0}),$$

$$(U_{l(n)+2|r_0^1|}, H_k(U_n^2, R_{l(n)}^1), ..., H_k(U_n^2, R_{l(n)}^n), H_k(U_n^3, R_{l(n)}^{n+1}), ..., H_k(U_n^3, R_{l(n)}^{2n})) < \mu(n) \tag{4.8}$$

$$\implies \Delta(O_k^0(X_n, R_n^{O^0}), U_{2|r_0^1|+l(n)+2nl(l(n))}) < \mu(n) \tag{4.9}$$

$$\implies \Delta(O_k^0(X_n, R_n^{O^0}),$$

$$(U_{2|r_0^1|} + l(n), H_k(H_k(X_n, R_n), R_{l(n)}^1), ..., H_k(H_k(X_n, R_n), R_{l(n)}^{2n}))) < \mu(n) \tag{4.10}$$

$$\implies \Delta(O_k'(X_n, R_n^{O_1}), U_{2|r_0^1|+l(n)+nl(l(n))}) < \mu(n) \tag{4.11}$$

Eq. 4.7 follows from statistical pseudorandomness of $\mathbb{H}$. Using again statistical pseudorandomness and the fact that for any function ϕ and any two distributions X and Y, $\Delta(\phi(X_n), \phi(Y_n)) \leq \Delta(X_n, Y_n)$, we get Eq. 4.8 (here, the function ϕ_1 takes three strings, $(r_0^1, y_1), (r_0^2, y_2), (r_0^3, y_3)$, as input and outputs $r_0^2, r_0^3, (r_0^1, y_1)$, n images of the second string and then n images of the third one). Eq. 4.9 and 4.10 are true because $\mathbb{H}$ is statistically perfectly one-way. To prove Eq. 4.11, we use the previous fact ($\Delta(\phi(X_n), \phi(Y_n)) \leq \Delta(X_n, Y_n)$) by having a function, ϕ_2, that takes $O_k^0(x, .)$, converts it to $O_k^n(x, .)$, and then truncates it to $O_k'(x, .)$. Note that if ϕ_2 is applied to the second distribution in Eq. 4.10, then it yields the same distribution. Since this distribution is statistically close to uniform ($\mathbb{H}$ is statistically pseudorandom), then triangle inequality implies that $O_k'(x, .)$ is statistically close to uniform.

To finish the proof of the theorem, we use the assumption that G is computational pseudorandom with auxiliary information to conclude that $O_k(X_n, R_n^O) = O_k'(X_n, R_n^{O'}), G_k(X_n, R_n^G)$ is computationally indistinguishable from $O_k'(X_n, R_n^{O'}), U_{l(n)}$, and therefore, $O_k(X_n, R_n^O)$ is, by the pseudorandom property of O_k', computationally

indistinguishable from uniform. □

Collision resistance and public randomness. If both $\mathbb{H}$ and $\mathbb{G}$ have public randomness then $\mathbb{O}$ inherits this property. Also, $\mathbb{O}$ is collision resistant if either $\mathbb{H}$ or $\mathbb{G}$ is.

4.3.2.2 Extractable POW Functions with Auxiliary Information

Theorem 4.3.3 claims that Construction 4.3.2 is a POW function. However, it is not known whether this construction is perfectly one-way with auxiliary information. To obtain this property, it seems we need to assume, in addition, that $\mathbb{H}$ is computationally perfectly one-way with auxiliary information. Alternatively, we use a simpler assumption to prove perfect one-wayness with auxiliary information of the following construction.

Construction 4.3.3. *Let $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$ be a verifiable family ensemble, where $H_k : \{0,1\}^n \times R_n \rightarrow \{0,1\}^{l(n)}$ for some polynomial l. Denote by $\mathbb{O} = \{\mathbf{O}^n\}_{n\in\mathbb{N}}$, where $O_k : \{0,1\}^n \times R_n^O = (R_{n+1}^{(2n)}) \rightarrow \{0,1\}^{2nl(n+1)}$, the family ensemble defined as:*

$$O_k(\ x,\ (r_1...,r_{2n})\ ) =$$

$$H_k((x,1),r_1),...,H_k((x,1),r_n),\ H_k((x,x_1),r_{n+1}),...,H_k((x,x_n),r_{2n}),$$

where x_i is the i^{th} bit of x.

The verifier, $V_{\mathbb{O}}$, for this construction accepts its input, $x, y = (y_1, ..., y_{2n})$, if and only if $V_{\mathbb{H}}((x,1),y_1) = ... = V_{\mathbb{H}}((x,1),y_n) = V_{\mathbb{H}}((x,x_1),y_{n+1}) = ... = V_{\mathbb{H}}((x,x_n),y_{2n}) = 1$.

Perfect one-wayness. To prove that Construction 4.3.3 is perfectly one-way with auxiliary input, we need to assume that an adversary can not distinguish images of input strings ending in 1 from those ending in 0 even in the presence of auxiliary information. We do not know whether this assumption is implied by conventional perfectly one-way definitions. However, we present a generalization of known indistinguishability definitions that satisfies the aforementioned assumption. Like this assumption, this generalized definition is not known to be implied by conventional perfectly one-way definitions. However, it seems to be a natural generalization of indistinguishability. In more detail, some of the existing perfectly one-way definitions require that images of a polynomial number of input strings taken from well-spread distributions are indistinguishable

from uniform, even in the presence of auxiliary information. In this case, the well-spread distributions are assumed to be independent of one another. Alternatively, other existing definitions specify a fixed relation between these distributions, e.g., all input strings are the same. The new definition combines these two notions to require indistinguishability with respect to *any* vector of polynomially-many well-spread distributions. We highlight that the difference between this definition and the existing ones is that even though each distribution is *individually* well-spread, these distributions, taken together, may be arbitrarily correlated. For instance, for proving that Construction 4.3.3 is a POW function, we require that $\mathbb{H}$ is a POW function with respect to a vector of $2n$ well-spread distributions. This vector of distributions has a sampling algorithm that outputs $2n$ input strings of the form $x'_1 = (x, 1), ..., x'_n = (x, 1), x'_{n+1} = (x, x_1), ..., x'_{2n} = (x, x_n)$, where x is sampled from a well-spread distribution and x_i is the ith bit of x. Formally,

Definition 4.3.5 (Strong Pseudorandomness). *A verifiable family ensemble, $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$, where $H_k : \{0,1\}^n \times R_n \to \{0,1\}^{l(n)}$ for some polynomial l, is called **strongly pseudorandom with auxiliary information** if for any vector of polynomially-many well-spread distributions, $\mathbb{X} = \{X^1, ..., X^t\}$, with sampling algorithm, G, any uninvertible function in t variables, F, any PPT, A, and any $k \in K_n$:*

$$|Pr[(x_1, ..., x_{t(n)}) \leftarrow G(1^n),\ z \leftarrow F(x_1, ..., x_{t(n)}),\ r_1, ..., r_{t(n)} \leftarrow R_n, ..., R_n,$$

$$b \leftarrow A(k, z, H_k(x_1, r_1), ..., H_k(x_{t(n)}, r_{t(n)})) : b = 1] -$$

$$Pr[(x_1, ..., x_{t(n)}) \leftarrow G(1^n),\ z \leftarrow F(x_1, ..., x_{t(n)}),\ b \leftarrow A(k, z, U_{l(|x_1|)+...+l(|x_{t(n)}|)}) : b = 1]|$$

$$\leq \mu(n)$$

We remark that this definition is of independent interest. In particular, it is also used in Chapter 7 for instantiating Random Oracle in first-query-hiding encryption schemes.

Theorem 4.3.4. *If a family ensemble, $\mathbb{H}$, satisfies Definition 4.3.5, then Construction 4.3.3 is computationally pseudorandom with auxiliary information (as in Definition 2.5.6).*

Proof. For simplicity, we prove that $\mathbb{O}$ is computationally 1-pseudorandom with auxiliary information. Proof of the general case is similar. This proof is straightforward. The

output of $\mathbb{O}$ consists of $2n$ images under $\mathbb{H}$. By the assumption on $\mathbb{H}$, these $2n$ images are indistinguishable from uniform even in the presence of auxiliary information. Formally, let $\mathbb{X}$ be any well-spread distribution. We define a vector of $2n$ well-spread distributions, $\mathbb{X}' = \{X'^1, ..., X'^{2n}\}$. The sampling algorithm, G, for $\mathbb{X}'$ outputs a $2n$-tuple of the form $x'_1 = (x, 1), ..., x'_n = (x, 1), x'_{n+1} = (x, x_1), ..., x'_{2n} = (x, x_n)$, where x is sampled from $\mathbb{X}$ and x_i is the ith bit of x. It is easy to see that any x'_i is taken from a well-spread distribution. Moreover, any uninvertible function, F, in x is also uninvertible in $x'_1, ..., x'_{2n}$. Let F' be the function that takes $x'_1, ..., x'_{2n}$ as input and computes $F(x)$. Thus, for any adversary, A:

$$Pr[x \leftarrow X_n,\ z \leftarrow F(x),\ r^O \leftarrow R_n^O,\ b \leftarrow A(k, z, O_k(x, r^O)) : b = 1] =$$

$$Pr[(x'_1, ..., x'_{2n}) \leftarrow G(1^n),\ z \leftarrow F'(x_1, ..., x_{2n}),\ r_1, ..., r_{2n} \leftarrow R_n, ..., R_n,$$

$$b \leftarrow A(k, z, H_k(x_1, r_1), ..., H_k(x_{2n}, r_{2n})) : b = 1]$$

Using the assumption that $\mathbb{H}$ satisfies Definition 4.3.5, the theorem follows.

□

Extraction. Extraction is very similar to that of Construction 4.3.2. As before, the extractor forces the adversary to output two related images from which it can recover a preimage.

However, the disadvantage of this construction is that strong collision resistant may not be sufficient. In particular, it may be the case that there is some x for which $H_k((x, 1), r) = H_k((x, 0), r)$ for any r. Moreover, there may be an adversary that successfully computes $O_k(x)$ for any r. This case does not contradict strong collision resistance because every image of $x, 1$ is also an image of $x, 0$. Also, it may be hard to find such an x (to contradict collision resistance). Consequently, we assume that $\mathbb{H}$ is injective (as in Definition 4.3.6). In Section 4.3.2.3, we show how to realize injection from strong POW functions that have a "weak" collision resistance property.

Definition 4.3.6 (Injective family ensembles). *A verifiable family ensemble, $\mathbb{H}$, is called injective, if for any k and any y, there exists at most one x such that $V_{\mathbb{H}}(x, y) = 1$.*

Theorem 4.3.5. *If $\mathbb{H}$ is a verifiable and injective family ensembles (as in Definition*

4.3.6), then Construction 4.3.3 is blackbox extractable with auxiliary information (as in Definition 4.2.6).

Proof. This proof is very similar to that of Theorem 4.3.2. Again, we give a black-box extractor, $\mathcal{K}$, that works with probability polynomially close to 1. This extractor also receives a polynomial bound, which represents the allowed margin of error, and halts in time polynomially related to this bound.

Formally, $\mathcal{K}$ works as follows:

input : k, p
interaction: with an external PPT, A

receive y_0;
for $j = 1$ **to** n **do**
  $r_1, ..., r_n \leftarrow R_n, ..., R_n$;
  for $i = 1$ **to** $16n^3p^2(n)$ **do**
    $r_{n+1}, ..., r_{2n} \leftarrow R_n, ..., R_n$;
    $u_1, ..., u_n \leftarrow R_n, ..., R_n$;
    $r_1^O = r_1, ..., r_{2n}$;
    $u_1^O = u_1, ..., u_n, r_1, ..., r_n$;
    $r_2^O, ..., r_n^O \leftarrow R_n^O, ..., R_n^O$;
    $u_2^O, ..., u_n^O \leftarrow R_n^O, ..., R_n^O$;
    $d_1, d_2 \leftarrow \{1, ..., n\}, \{1, ..., n\}$;
    send $r_2^O, ..., r_{d_1}^O, r_1^O, r_{d_1+1}^O, ..., r_n^O$;
    receive $y_1, ..., y_n$;
    rewind A;
    send $u_2^O, ..., u_{d_2}^O, u_1^O, u_{d_2+1}^O, ..., u_n^O$;
    receive $v_1, ..., v_n$;
    rewind A;
    parse y_{d_1} **as** $y_{d_1}^1, ..., y_{d_1}^{2n}$;
    parse v_{d_2} **as** $v_{d_2}^1, ..., v_{d_2}^{2n}$;
    for $k = 1$ **to** n **do**
      if $y_{d_1}^k = v_{d_2}^{n+k}$ **then**
        $x_i = 1$;
      else
        $x_i = 0$;
    end
    $x = x_1, ..., x_n$;
    if $V_{\mathbb{O}}(x, y_0) = 1$ **and** $\forall i \geq 1,\ y_i = O_k(x, r_i^O)$ **and** $\forall i \geq 1,\ v_i = O_k(x, u_i^O)$ **then**
      return x;
  end
end
$x \leftarrow U_n$;
return x;

Algorithm 4.3.3: $\mathcal{K}$

Analysis

As in the proof of Theorem 4.3.2, we show that if for some k, z, r_A and some polynomial, p, we have:

$$Pr[(r_1^O, ..., r_n^O) \leftarrow (R_n^O, ..., R_n^O),\ A(k, z, r_A) = (y_0, s),\ A(s, r_1^O, ..., r_n^O) = (y_1, ..., y_n) :$$

$$\exists x',\ \forall i,\ y_i = O_k(x', r_i^O)] \geq \frac{1}{p(n)}. \tag{4.12}$$

then, for the same k, z, r_A:

$$Pr[A(k, z, r_A) = (y_0, s),\ x \leftarrow \mathcal{K}^A(k, p) : V_{\mathbb{O}}(x, y_0) = 1] > 1 - \mu(n). \tag{4.13}$$

Recall that each r_i^O is taken from $R_n^O = R_{n+1}^{2n}$. Without loss of generality, assume that $R_{n+1} = \{0,1\}^n$.

Eq. 4.12 can be rephrased as: there exists a subset $S \subseteq \underbrace{(R_n^O, ..., R_n^O)}_{n}$, $|S| \geq \frac{2^{2n^3}}{p(n)}$ such that for all $(r_1^O, ..., r_n^O) \in S$,$A(k, z, r_A) = (y_0, s)$, $A(s, r_1^O, ..., r_n^O) = (y_1, ..., y_n)$, and $\exists x',\ \forall i,\ y_i = O_k(x', r_i^O)$ and $V_{\mathbb{O}}(x', y_0) = 1$.

Denote by a_i^O (respectively, b_i^O) the first n (respectively, last n) strings of r_i^O, i.e., $r_i^O = a_i^O, b_i^O$. Now, for each $r \in \{0,1\}^{n^2}$, let t_r^1 (respectively, t_r^2) denote the number of times that r appears as a^O (respectively, b^O) in any r_i^O in any vector in S. Furthermore, let S' be a new set that contains all the vectors in S except those that contain an r as a^O with $t_r^1 \leq \frac{2^{2n^3}}{2^{n^2} 4p(n)}$ or r as b^O with $t_r^2 \leq \frac{2^{2n^3}}{2^{n^2} 4p(n)}$. Since for each r at most $\frac{2^{2n^3}}{2^{n^2} 2p(n)}$ vectors are deleted from S and there are at most 2^{n^2} such r, $|S'| \geq \frac{2^{2n^3}}{2p(n)}$. Now, let T_1 (respectively, T_2) be the set of elements that occur as a^O (respectively, b^O) in any r_i^O in any vector in S'. Let $T = T_1 \cap T_2$. We should have:

$$|T_1|^n |T_2|^n \geq |S'| \geq \frac{2^{2n^3}}{2p(n)}$$

$$\implies |T_1||T_2| \geq \frac{2^{2n^2}}{(2p(n))^{\frac{1}{n}}} \geq 2^{2n^2 - \frac{clogn}{n}}$$

$$\implies |T_1| \geq 2^{n^2 - \frac{clogn}{n}} \ and \ |T_2| \geq 2^{n^2 - \frac{clogn}{n}}$$

$$\implies |T| > \frac{2^{n^2}}{2},$$

where c is some constant that depends on p, and the last inequality holds for sufficiently

large n.

Therefore, the probability that $\mathcal{K}$ does not find, in line 3 of its code and for all n repetitions, $r_1, ..., r_n$ that belongs to T is less than $\frac{1}{2^n}$. Now, suppose that $r_1, ..., r_n$ happens to be in T. The probability that a uniformly chosen vector $r_1^O, ..., r_n^O$, conditioned on containing $r_1, ..., r_n$ as a^O in some r_i^O, falls in S is:

$$Pr[(r_1^O, ..., r_n^O) \leftarrow R_n^O \times ... \times R_n^O : (r_1^O, ..., r_n^O) \in S | \exists i \text{ and } b^O : r_i^O = (a^O, b^O)]$$

$$\geq \frac{\frac{2^{2n^3}}{2^{n^2} 4p(n)}}{\frac{n 2^{2n^3}}{2^{n^2}}} = \frac{1}{4np(n)}$$

The same inequality holds for b^O.

So that, if $\mathcal{K}$ finds, in line 3 of the code and in some iteration, an $r_1, ..., r_n$ that belongs to T, the probability that both vectors $r_1^O, ..., r_n^O$ and $u_1^O, ..., u_n^O$, computed in lines $5 - 10$, are in S is $\frac{1}{(4np(n))^2}$. Repeated sampling for $16n^3p^2(n)$ times ensures that the probability of failure in all $16n^3p^2(n)$ iterations is negligible.

Now, if both vectors $r_1^O, ..., r_n^O$ and $u_1^O, ..., u_n^O$ are in S, then by definition of S, A will compute consistent $y_1, ..., y_n$ and $v_1, ..., v_n$ in lines $13 - 16$. In other words, $\exists x_1,\ V_{\mathbb{O}}(x_1, y_0) = 1$, and $\forall i,\ y_i = O_k(x_1, r_i^O)$, and $\exists x_2,\ V_{\mathbb{O}}(x_2, y_0) = 1$, and $\forall i,\ v_i = O_k(x_2, u_i^O)$. However, by injection, $x_1 = x_2 = x'$. Now, we need to show that x as computed in line 26 is equal to x'. Observe that if the ith bit of x' is 1 then $x_i = 1$ because $y_{d_1}^i = H_k((x', 1), r) = H_k((x', x_i'), r) = v_{d_2}^{n+i}$ (line 21). On other hand, if $x_i' = 0$, then x_i is also 0 because $v_{d_2}^{n+i} = H_k((x', x_i'), r) = H_k((x', 0), r)$, which by injection and efficient verification is not equal to $H_k((x', 1), r) = y_{d_1}^i$.

Using injection, x, as computed by $\mathcal{K}$, is a common preimage of $y_1, ..., y_n$. Combining Eq. 4.12 and 4.13 with the last claim finishes the proof.

□

Collision resistance and public randomness. Construction 4.3.3 inherits public randomness, collision resistance, and injection from the underlying primitive in a straightforward way.

4.3.2.3 Injective POW Functions from Strong Perfect One-wayness

Theorems 4.3.2 and 4.3.5 use stronger assumptions than conventional collision resistance, namely strong collision resistance and injection. So, we study the feasibility of such assumptions. In particular, we show that injection can be achieved from strong perfect one-wayness and encryption schemes. In more detail, we use perfect completeness of encryption schemes (i.e., for all pk, sk and all messages m, $D_{sk}(E_{pk}(m)) = m$) as a way of achieving injection while maintaining a certain level of secrecy via semantic security. The formal construction follows.

Construction 4.3.4. *Let $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$ be a verifiable family ensemble, where for some polynomial $l(n)$, for any $n \in \mathbb{N}$, and any $k \in K_n$, $H_k : \{0,1\}^n \times R_n \to \{0,1\}^{l(n)}$ and (G, E, D) be a public-key encryption scheme. Denote by $\mathbb{O} = \{\mathbf{O}^n\}_{n\in\mathbb{N}}$, the family ensemble defined as:*

$$O_k(x, (r_1, ..., r_{|r^E|}, pk, r^E)) =$$

$$\begin{cases} x, & \text{if } \exists i, H_k((x,0), r_i) = H_k((x,1), r_i) \\ pk, H_k((x, r_1^E), r_1), ..., H_k((x, r_{|r^E|}^E), r_{|r^E|}), E_{pk}(x, r^E) & \text{otherwise,} \end{cases}$$

where r_i^E is the ith bit of r^E.

Construction 4.3.4 has the following verifier. $V_{\mathbb{O}}(x, y) = 1$ if and only if $y = x$ (case 1) or $\forall i$, either $V_{\mathbb{H}}((x,0), y_i) = 1$ or $V_{\mathbb{H}}((x,1), y_i) = 1$ and $E_{pk}(x, r^E) = c$ (case 2), where $y = pk, y_1, ..., y_{|r^E|}, c$ and $r^E = V_{\mathbb{H}}((x,1), y_1), ..., V_{\mathbb{H}}((x,1), y_{|r^E|})$.

Injection. Disregarding the first case for now, this construction achieves injection due to perfect completeness of the encryption scheme. Informally, $V_{\mathbb{O}}$ recovers r^E from x and the $\mathbb{H}$ images in the output of $\mathbb{O}$. Once r^E is found, it is easy to recompute $E_{pk}(x, r^E)$ and check the validity of the last string of in the output of $\mathbb{O}$.

However, the second case by itself is not sufficient to guarantee both injection and verification. To see why, note that the constant function is a verifiable strongly POW function. For such functions, we can not recover r^E in the second case of Construction 4.3.4 because $H_k((x,1), r) = H_k((x,0), r)$ for any x and r. So, either $V_{\mathbb{O}}$ will not accept $x, H(x)$, compromising verification, or it will accept any input, compromising injection. To solve this problem, we introduce the first case to the construction. Now, if a collision,

of the form described above, occurs, case 1 will be used. Thus, the two cases combined guarantee injection.

Theorem 4.3.6. *If $\mathbb{H}$ is a verifiable family ensemble and (G, E, D) is an asymmetric encryption scheme with perfect completeness (as in Definition 2.7.1), then Construction 4.3.4 is both verifiable and injective (as in Definition 4.3.6).*

Proof. **Verification.** For any k, x, and r^O, $V_{\mathbb{O}}(x, O_k(x, r^O)) = 1$: If $O_k(x, r^O) = x$, then $V_{\mathbb{O}}$ accepts immediately. If $O_k(x, r^O) \neq x$ then we know that $O_k(x, r^O) = pk, y_1, ..., y_{|r^E|}, c$ and for any i, $\exists!$ b such that $V_{\mathbb{H}}((x, b), y_i) = 1$. Thus, $V_{\mathbb{O}}$ recovers r^E, computes $E_{pk}(x, r^E)$, and accepts (because $E_{pk}(x, r^E) = c$ and for any i, $\exists!$ b such that $V_{\mathbb{H}}((x, b), y_i) = 1$) .

Injection. W.l.o.g. assume that the input domain of $\mathbb{O}$ is $\{0,1\}^n$ and the range of the second case of $\mathbb{O}$ is a subset of $\{0,1\}^{l(n)}$, with $l(n) > n$. For any k and y, if $|y| = n$ (case 1) then there exists exactly one input x (namely, $x = y$), such that $V_{\mathbb{O}}(x, y) = 1$. On the other hand, if $y = pk, y_1, ..., y_{|r^E|}, c$ and if there are two inputs x_1 and x_2, such that $V_{\mathbb{O}}(x_1, y) = V_{\mathbb{O}}(x_2, y) = 1$, then there are r^E and u^E such that $E_{pk}(x_1, r^E) = E_{pk}(x_2, u^E) = c$. By perfect completeness, $x_1 = x_2$.

□

Perfect One-wayness. As we mentioned in the previous paragraph, the class of strong POW functions contains some trivial ones such that the constant functions. If such functions are used in Construction 4.3.4, it does not provide secrecy (in fact, it outputs x in the clear). For Construction 4.3.4 to be secure, the underlying function, $\mathbb{H}$, should be nontrivial. Specifically, $\mathbb{H}$ should "preserve" entropy of the input. In particular, we require that $H_k((x, 1), r) \neq H_k((x, 0), r)$ for input, x, with sufficiently high entropy, and a uniformly chosen r. Note that this property is implied by collision resistance. In fact, it is much weaker than collision resistance: pick any function with collision resistance and modify it so that it has the same output on 0^n and 1^n. The formal definition of entropy preservation follows.

Definition 4.3.7 (Weak Entropy Preservation). *A family ensemble, $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$, where $H_k : \{0,1\}^n \times R_n \to \{0,1\}^{l(n)}$ for some polynomial l, **weakly preserves entropy***

if for any well-spread distribution, $\mathbb{X}$, and any $k \in K_n$:

$$|Pr[x \leftarrow X_n,\ r \leftarrow R_n : H_k((x,0),r) = H_k((x,1),r))] \leq \mu(n)$$

A possible disadvantage of Construction 4.3.4 is that its secrecy depends on the secrecy of both $\mathbb{H}$ and the encryption scheme. Specifically, whether it is pseudorandom or not depends on whether both $\mathbb{H}$ and the encryption scheme are pseudorandom or not. By a pseudorandom encryption scheme, we mean that a ciphertext of a message taken from a well-spread distribution is indistinguishable from a uniform string. Such encryption schemes are known to exist, e.g., the encryption scheme in [Can97], (which is an instantiation of an encryption scheme in the RO model that appeared in [BR93]). However, if the encryption scheme is not pseudorandom, $\mathbb{O}$ still satisfies computational indistinguishability. For completeness, we present here the definition of strong indistinguishability.

Definition 4.3.8 (Strong Indistinguishability). *A verifiable family ensemble, $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$, where $H_k : \{0,1\}^n \times R_n \to \{0,1\}^{l(n)}$ for some polynomial l, is called **strongly indistinguishability with auxiliary information** if for any vector of polynomially-many well-spread distributions, $\mathbb{X} = \{X^1, ..., X^t\}$, with sampling algorithm, G, any uninvertible function in t variables, F, any PPT, A, and any $k \in K_n$:*

$$|Pr[(x_1, ..., x_{t(n)}) \leftarrow G(1^n),\ z \leftarrow F(x_1, ..., x_{t(n)}),\ r_1, ..., r_{t(n)} \leftarrow R_n, ..., R_n,$$

$$b \leftarrow A(k, z, H_k(x_1, r_1), ..., H_k(x_{t(n)}, r_{t(n)})) : b = 1] -$$

$$Pr[(x_1, ..., x_{t(n)}) \leftarrow G(1^n),\ z \leftarrow F(x_1, ..., x_{t(n)}),\ r_1, ..., r_{t(n)} \leftarrow R_n, ..., R_n,$$

$$u_1, ..., u_{t(n)} \leftarrow U_n, ..., U_n,\ b \leftarrow A(k, z, H_k(u_1, r_1), ..., H_k(u_{t(n)}, r_{t(n)})) : b = 1]| \leq \mu(n)$$

Theorem 4.3.7. *If $\mathbb{H}$ is strongly pseudorandom with auxiliary information (as in Definition 4.3.5) and preserves entropy (as in Definition 4.3.7) and (G, E, D) is a semantically-secure encryption scheme (respectively, pseudorandom encryption scheme), then Construction 4.3.4 is strongly indistinguishable as in Definition 4.3.8 (respectively, pseudorandom as in Definition 4.3.5) with auxiliary information.*

Proof. Let A be any PPT. Denote by $|r^E|$ the length of the randomness for E. For any

vector of t well-spread distributions, $\mathbb{X}$, with sampling algorithm, S, let $\mathbb{X}'$ be a vector of $|r^E|t$ well-spread distributions with a sampler, S'. S' runs S to sample $x_1, ..., x_t$. It then outputs $(x_1, b_1^1), ..., (x_1, b_1^{|r^E|}), ..., (x_t, b_t^1), ..., (x_t, b_t^{|r^E|})$, where all the b_i^j are random bits. Moreover, for any uninvertible function, F in t parameters, $x_1, ..., x_t$, let F' be the function in $|r^E|t$ parameters, $(x_1, b_1^1), ..., (x_1, b_1^{|r^E|}), ..., (x_t, b_t^1), ..., (x_t, b_t^{|r^E|})$, that samples $pk_1, ..., pk_t$, and outputs

$$pk_1, ..., pk_t, F(x_1, ..., x_t), E_{pk_1}(x_1, (b_1^1, ..., b_1^{|r^E|})), ..., E_{pk_t}(x_t, (b_t^1, ..., b_t^{|r^E|})).$$

By semantic security and uninvertibility of F, F' is uninvertible.

By entropy preservation, we have for the same parameters:

$$|Pr[(x_1, ..., x_{t(n)}) \leftarrow S(1^n),\ z \leftarrow F(x_1, ..., x_{t(n)}),\ r_1, ..., r_{t(n)} \leftarrow R_n^O, ..., R_n^O,$$

$$b \leftarrow A(k, z, O_k(x_1, r_1), ..., O_k(x_{t(n)}, r_{t(n)})) : b = 1] -$$

$$Pr[(x_1, ..., x_{t(n)}) \leftarrow S(1^n),\ z \leftarrow F(x_1, ..., x_{t(n)}),\ r_1, ..., r_{t(n)} \leftarrow R_n^O, ..., R_n^O,$$

$$b \leftarrow A(k, z, O_k(x_1, r_1), ..., O_k(x_{t(n)}, r_{t(n)})) : b = 1 | \forall i, O_k(x_i, r_i) \neq x_i]| \leq \mu(n).$$

Moreover, we have by definition:

$$Pr[(x_1, ..., x_{t(n)}) \leftarrow S(1^n),\ z \leftarrow F(x_1, ..., x_{t(n)}),\ r_1, ..., r_{t(n)} \leftarrow R_n^O, ..., R_n^O,$$

$$b \leftarrow A(k, z, O_k(x_1, r_1), ..., O_k(x_{t(n)}, r_{t(n)})) : b = 1 | \forall i, O_k(x_i, r_i) \neq x_i]$$

$$= Pr[(x_1, b_1^1), ..., (x_{t(n)}, b_{t(n)}^{|r^E|}) \leftarrow S'(1^n),\ z' \leftarrow F'((x_1, b_1^1), ..., (x_{t(n)}, b_{t(n)}^{|r^E|})),$$

$$r_1, ..., r_{t(n)|r^E|} \leftarrow R_n, ..., R_n,$$

$$b \leftarrow A(k, z', H_k((x_1, b_1^1), r_1), ..., H_k((x_{t(n)}, b_{t(n)}^{|r^E|}), r_{t(n)|r^E|})) : b = 1]$$

Since $\mathbb{H}$ is strongly pseudorandom (or strongly indistinguishable, if we are considering this case), we have:

$$|Pr[(x_1, b_1^1), ..., (x_{t(n)}, b_{t(n)}^{|r^E|}) \leftarrow S'(1^n),\ z' \leftarrow F'((x_1, b_1^1), ..., (x_{t(n)}, b_{t(n)}^{|r^E|})),$$

$$r_1, ..., r_{t(n)|r^E|} \leftarrow R_n, ..., R_n,$$

$$b \leftarrow A(k, z', H_k((x_1, b_1^1), r_1), ..., H_k((x_{t(n)}, b_{t(n)}^{|r^E|}), r_{t(n)|r^E|})) : b = 1] -$$

$$Pr[(x_1, b_1^1), ..., (x_{t(n)}, b_{t(n)}^{|r^E|}) \leftarrow S'(1^n),\ z' \leftarrow F'((x_1, b_1^1), ..., (x_{t(n)}, b_{t(n)}^{|r^E|})),$$

$$b \leftarrow A(k, z', U_{|r^E|t(n)l(n+1)})) : b = 1]| \leq \mu(n)$$

Since (G, E, D) is pseudorandom (or semantic security, if we are considering this case), we have:

$$|Pr[(x_1, b_1^1), ..., (x_{t(n)}, b_{t(n)}^{|r^E|}) \leftarrow S'(1^n),\ z' \leftarrow F'((x_1, b_1^1), ..., (x_{t(n)}, b_{t(n)}^{|r^E|})),$$

$$b \leftarrow A(k, z', U_{|r^E|t(n)l(n+1)})) : b = 1] -$$

$$Pr[(x_1, ..., x_{t(n)}) \leftarrow S(1^n),\ z \leftarrow F(x_1, ..., x_{t(n)}),\ b \leftarrow A(k, z, U_{t(n)|E_{pk}(x_1)|}, U_{|r^E|t(n)l(n+1)})) :$$

$$b = 1]| \leq \mu(n)$$

□

Using the fact that entropy preservation is implied by collision resistance, we have the following corollary.

Corollary 4.3.1. *If there exists a strong pseudorandom POW function with collision resistance and a pseudorandom encryption scheme, then there exists an injective, strong pseudorandom POW function.*

4.4 On the Connection to Σ−Protocols

We show how Σ-protocols (see Definition 2.12.1) and POW functions can be used to construct *another variant* of extractable POW functions. We discuss the differences between this construction and the original definition after presenting the construction. At a high level, we use a POW function, $\mathbb{H}$, and a Σ-protocol for the language consisting of the range of $\mathbb{H}$. The new function, $\mathbb{O}$ incorporates the prover messages in its output. To do so, $\mathbb{O}$ has an additional random coin that determines which one of the two prover messages it outputs. Formally,

Construction 4.4.1. *Let $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$ be a verifiable family ensemble and $L_{\mathbb{H}}$ be the family of languages $L_{H_k} = \{y : \exists x, r, H_k(x,r) = y\}$. Let $\mathcal{P} = (P, V)$ be a Σ-protocol for L_{H_k}. Recall that e is the random string V sends in the second round. Denote by $m_0(y,(x,r),r_P)$ and $m_1(y,(x,r),e,r_P)$ the messages sent by P (with private coins r_P) in the first and third round. Then, let $\mathbb{O}$ be the following family ensemble:*

$$O_k(x,r,r_P,e,b) = \begin{cases} H_k(x,r), e, m_0(H_k(x,r),(x,r),r_P), & \text{if } b = 0 \\ H_k(x,r), e, m_1(H_k(x,r),(x,r),e,r_P), & \text{otherwise} \end{cases} \tag{4.14}$$

We associate with $\mathbb{O}$ a verifier, $V_{\mathbb{O}}$. $V_{\mathbb{O}}$ simulates $V_{\mathbb{H}}$ on $(x,H_k(x,r))$ that is $V_{\mathbb{O}}(x, y = (y_1, e, m_b)) = V_{\mathbb{H}}(x, y_1)$.

Extraction. Construction 4.4.1 inherits a special form of extraction from the Σ-protocol. Specifically, the challenger plays the role of V in $\mathcal{P}$. A starts the game by sending an image of x, the challenger responds with e, and A sends another image using e as public coins for $\mathcal{P}$. In more detail, A sends $O_k(x,r,r_P,e_0,0) = H_k(x,r), e_0, m_0$ in the first round and $O_k(x,r,r_P,e_1,1) = H_k(x,r), e_1, m_1$ in the third one (see Figure 4.2). We emphasize that r, r_P is the same in the first and third round and e_1 is chosen by the challenger. The interaction is called consistent if V accepts the conversation $(k, H_k(x,r), m_0, e_1, m_1)$. Then, extraction means A knows a preimage x if the interaction is consistent. The formal notion of this extraction follows with $\hat{V}$ defined in Algorithm 4.4.1.

Definition 4.4.1 (Σ-Extraction). *Let $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$ be a verifiable family ensemble, where $H_k : \{0,1\}^n \times R_n \to \{0,1\}^{l(n)}$ for some polynomial l. Then, $\mathbb{H}$ is called Σ-**extractable** if there exists a PPT, $\mathcal{K}$, such that for any distribution $\mathbb{Z} = \{Z_n\}_{n\in\mathbb{N}}$, any PPT A, any polynomial, p, and any $k \in K_n$:*

$$Pr[z \leftarrow Z_n,\ e \leftarrow U_n,\ (y_0, s) \leftarrow A(k,z),\ y_1 \leftarrow A(s,e), x \leftarrow \mathcal{K}^A(k,p) :$$

$$V_{\mathbb{H}}(x,y_0) = V_{\mathbb{H}}(x,y_1) = 1 \text{ or } \hat{V}(k,y_0,y_1,e) \neq 1)] > 1 - \frac{1}{p(n)} - \mu(n).$$

We show that Construction 4.4.1 satisfies Definition 4.4.1. Specifically, if $\hat{V}(k,y_0,y_1,e) = 1$, then the conversation is accepted by V. By the special soundness property on $\mathcal{P}$, $\mathcal{K}^A$ can access A (with rewinding) to extract a preimage. Formally,

A		Challenger
$y_0 = O_k(x, r_0, r_P, e_0, 0)$		
	$\xrightarrow{y_0}$	
		$e_1 \leftarrow U_n$
	$\xleftarrow{e_1}$	
$y_1 = O_k(x, r_0, r_P, e_1, 1)$		
	$\xrightarrow{y_1}$	

Figure 4.2: 3-round Interaction of Σ-extraction

input: $k, y_0 = (u_0, e_0, m_0), y_1 = (u_1, e_1, m_1), e$

if $u_0 = u_1$ **and** $e = e_1$ **and** $V(u_0, m_0, e, m_1) = 1$ **then**
    return 1;
else
    return 0;
end

Algorithm 4.4.1: $\hat{V}$

Theorem 4.4.1. *If $\mathbb{H}$ is a verifiable family ensemble and (P, V) is a family of Σ-protocols for the family of languages $L_{\mathbb{H}}$, then Construction 4.4.1 is Σ-extractable (as in Definition 4.4.1).*

Proof. The proof follows similar lines as the proofs of the last two extraction theorems. The universal extractor, $\mathcal{K}$, is defined in Algorithm 4.4.2, where V is the verifier from the Σ-protocol and $\mathcal{K}_{(P,V)}$ is the witness extractor given by the special soundness property.

input : k, p
interaction: with an external PPT, A

receive $y_0 = (u_0, e_0, m_0)$;
for $j = 1$ **to** $np^2(n)$ **do**
    $e_1 \leftarrow U_n$;
    $e_2 \leftarrow U_n$;
    send e_1;
    receive $y_1 = (u_1, e_1', m_1)$;
    rewind A;
    send e_2;
    receive $y_2 = (u_2, e_2', m_2)$;
    rewind A;
    if $u_0 = u_1 = u_2$ **and** $e_1' = e_1$ **and** $e_2' = e_2$ **and** $V(u_0, m_0, e_1, m_1) = 1$ **and** $V(u_0, m_0, e_2, m_2) = 1$ **then**
        $(x, r) = \mathcal{K}_{(P,V)}(u_0, (m_0, e_1, m_1), (m_0, e_2, m_2))$;
        return x;
end
$x \leftarrow U_n$;
return x;

Algorithm 4.4.2: $\mathcal{K}$

Analysis

Suppose that for some k, z, r_A and some polynomial, p, we have:

$$Pr[e \leftarrow U_n,\ (y_0, s) = A(k, z, r_A),\ y_1 = A(s, e) :$$

$$\hat{V}(k, y_0, y_1, e) = 1] \geq \frac{1}{p(n)}. \tag{4.15}$$

Then, we show that for the same k, z, r_A:

$$Pr[A(k, z, r_A) = (y_0, s),\ x \leftarrow \mathcal{K}^A(k, p) : V_{\mathbb{O}}(x, y_0)] > 1 - \mu(n). \tag{4.16}$$

In any iteration of the loop, the probability of the event that $\mathcal{K}$ does not find e_1, e_2, where $e_1 \neq e_2$ and A answers consistently on both e_1 and e_2 is at most $(1 - \frac{1}{p^2(n)}) + \mu(n)$. Thus, $\mathcal{K}$ does not find such a pair, e_1, e_2 in all $np^2(n)$ with probability at most $\mu(n)$, for sufficiently large n. On the other hand, if $\mathcal{K}$ finds such a good pair, it verifies this on line 11 and successfully extracts x using the witness extractor guaranteed by the special soundness property.

Moreover, by definition, we have $V_{\mathbb{O}}(x, y_1 = (u_1, e_1, m_1)) = V_{\mathbb{H}}(x, u_1)$ and since $u_0 = u_1$,$V_{\mathbb{H}}(x, u_1) = V_{\mathbb{H}}(x, u_0) = V_{\mathbb{O}}(x, y_0) = 1$. Thus, x is also a valid preimage for y_1.

Therefore, $\mathcal{K}$ fails with negligible probability except when Eq. 4.15 does not hold for some k, z, r_A. In the latter case, A is consistent no more than $\frac{1}{p(n)}$ of the time.

□

Perfect one-wayness. If $\mathbb{H}$ is an indistinguishable POW function, then, so is $\mathbb{O}$. However, the same statement does not hold for pseudorandomness unless the interaction in the Σ-protocol is computationally indistinguishable from uniform, when x has high min-entropy.

Informally, $\mathbb{O}$ is an indistinguishable POW function because given a sequence of images under $\mathbb{H}$, it is possible to convert them to a sequence of images of the same inputs under $\mathbb{O}$ (using the Σ-protocol simulator). Formally,

Theorem 4.4.2. *Let $\mathbb{H}$ be a verifiable t-indistinguishable POW function (respectively, with auxiliary information) as in Definition 2.5.5 and $\mathcal{P}_{\mathbb{H}}$ be a family of Σ-protocols*

for the family of languages $L_{\mathbb{H}}$ (as in Definition 2.12.1). Then, Construction 4.4.1 is t-indistinguishable (respectively, with auxiliary information) as in Definition 2.5.5.

Proof. For simplicity, we start with the case where $t = 2$ with auxiliary information. For any O_k with the language L_k, let (P, V) be the corresponding Σ-protocol with simulator S. Modify S into S' so that it outputs only one of the prover's messages, depending on an input b. Specifically, $S'(k, y_1, e, b) = y_1, e, m_b$, where $S(k, y_1, e) = m_1, e, m_2$. Observe that Σ-protocols retain the honest-verifier ZK property when proving two related theorem (specifically, $H_k(x, r_1), H_k(x, r_2)$) and in the presence of auxiliary information. So, by honest-verifier zero-knowledge, we have for any $(x, r_1, r_2, e_1, e_2, b_1, b_2)$: $z, O_k(x, r_1, ., e_1, b_1), O_k(x, r_2, ., e_2, b_2)$ has the same distribution as $z, S'(k, H_k(x, r_1), e_1, b_1), S'(k, H_k(x, r_2), e_2, b_2)$. By 2-indistinguishability on $\mathbb{H}$, we have for any well-spread distribution, $\mathbb{X}$, any k, and any uninvertible function, F:

$$|Pr[x \leftarrow X_n,\ z \leftarrow F(x),\ r_1, r_2 \leftarrow R_n, R_n,\ e_1, e_2 \leftarrow \{0,1\}^n, \{0,1\}^n,\ b_1, b_2 \leftarrow \{0,1\}^2 :$$

$$A(k, z, S'(k, H_k(x, r_1), e_1, b_1), S'(k, H_k(x, r_2), e_2, b_2)) = 1] \quad -$$

$$Pr[x \leftarrow X_n,\ z \leftarrow F(x),\ r_1, r_2 \leftarrow R_n, R_n,\ u_1, u_2 \leftarrow U_n, U_n,\ e_1, e_2 \leftarrow \{0,1\}^n, \{0,1\}^n,$$

$$b_1, b_2 \leftarrow \{0,1\}^2 : A(k, z, S'(k, H_k(u_1, r_1), e_1, b_1), S'(k, H_k(u_2, r_2), e_2, b_2)) = 1]| \leq \mu(n).$$

Using the honest verifier ZK property again, we have for any $(u_1, u_2, r_1, r_2, e_1, e_2, b_1, b_2)$: $z, S'(k, H_k(u_1, r_1), e_1, b_1), S'(k, H_k(u_2, r_2), e_2, b_2)$ has the same distribution as $z, O_k(u_1, r_1, ., e_1, b_1), O_k(u_2, r_2, ., e_2, b_2)$.

To prove the general case (for any polynomial t), observe that Σ-protocols retain the honest-verifier ZK property when proving multiple related theorem (in this case, the theorems are $H_k(x, r_1), ..., H_k(x, r_t)$). Then, the proof is similar to the previous one.

□

Collision resistance and public randomness. $\mathbb{O}$ inherits collision resistance from $\mathbb{H}$ in a straightforward way. However, $\mathbb{O}$ does not have public randomness even if $\mathbb{H}$ does.

4.4.1 Differences Among Constructions 4.3.2, 4.3.3, and 4.4.1

Definition 4.4.1 differs from Definition 4.2.6 in that $\mathbb{O}$ is not used in the three-round game as a function of x alone, rather as a function of x, r_P and the state of the protocol. Specifically, b is used to track the state of the protocol and the output of $\mathbb{O}$ depends on b because it contains m_b. On the other hand, we are able to achieve Σ-extraction using weaker assumptions than that of Constructions 4.3.2 and 4.3.3, namely, strong collision resistance and injection. So, it seems there is a tradeoff between the strength of the assumption used and the strength of the consistency requirement.

Moreover, Construction 4.4.1 achieves computational indistinguishability while the other constructions achieve the stronger pseudorandom property. Finally, unlike the first two construction, the Σ-construction does not inherit public randomness from the underlying function.

In conclusion, Σ-protocols allow extraction based on weaker assumptions for the price of maintaining state and weakening the secrecy requirement.

Chapter 5

Characterization of Extraction

Summary: We initiate a more general study of extractable functions, both interactive and noninteractive. This work is aimed at understanding the concept of extractability in of itself. In particular we demonstrate that a weak notion of extraction implies a strong one, and make rigorous the intuition that extraction and obfuscation are complementary notions.

5.1 Introduction

This chapter initiates a more general study of extractable functions, both interactive and noninteractive. Specifically, we address the following goal: understanding exactly what extraction means and how different notions of extraction (and lack of it) are related.

5.1.1 Our work

We attempt to address the question: What makes a function extractable? Moreover, if a function is extractable in a weak sense, does this mean that it is extractable in a strong sense? Towards answering these questions, we show that every function satisfies either a "weak" form of obfuscation [BGI+01] or a "weak" form of extraction. In other words, lack of extractability can be viewed as obfuscatability or resistance to "reverse engineering". This is indeed what one might naïvely expect - a function is either extractable or obfuscatable, and we show that this naïve thinking is correct to some extent. We then address the second question posed at the beginning of this paragraph. We find out that

This chapter is based on the paper [CD09], which is a joint work with Ran Canetti. Note that [CD09] contains some additional results that do not appear in this thesis.

for a large class of functions, notably, POW functions with auxiliary information, the answer to this question is positive.

5.1.1.1 Interactive Extraction

We discuss interactive extraction before noninteractive extraction. In this chapter, interactive extraction refers to the notion introduced in Chapter 4, except we require a single challenge instead of n.

On interactive extraction versus obfuscation. This line of work starts with an observation that extraction and obfuscation complement each other in a natural way. In other words, if a function is not extractable, then this lack of extractability is some form of obfuscation. Specifically, we call a function weakly (and interactively) extractable if for any adversary that is consistent in the interactive game with noticeable probability, there is a corresponding extractor that recovers a preimage with *noticeable success*. Moreover, the obfuscation mentioned previously relates to inability to "reverse engineer" an obfuscated program that produces images under the function. In other words, there is an obfuscated code that receives r as input and computes $f(x, r)$ for some x "hidden" in the obfuscated code. In more detail, we call f weakly obfuscatable if the following holds. There is an obfuscator that produces a program capable of correctly computing the function $f_x(r) = f(x, r)$ with noticeable probability, where x is chosen according to some well-spread distribution and then "hidden" in the program. Also, the program is considered obfuscated in the sense that it is hard to recover x from the obfuscated program, when x is drawn from the well-spread distribution mentioned above. The corresponding theorem can be stated in words as:

Theorem 5.2.1: Every family of probabilistic functions is either weakly extractable or weakly obfuscatable.

We emphasize that Theorem 5.2.1 is a general observation on any family of functions and does not assume anything about the family, not even that it is efficiently computable. Informally, this theorem can be argued for as follows. Suppose a function, f, is not weakly extractable. Then, there is an adversary A that answers consistently in the 3-round game of interactive extraction, and yet there is no extractor that recovers a preimage x. We use A to construct an obfuscation for the function f_x. The obfuscation simply contains the description of A and a corresponding private input that causes A to answer consistently.

To compute $f_x(r)$, simulate A, send r in the second round of the extraction game, and output the response of A. Functionality of this obfuscation follows from consistency of A while the hiding property follows directly from the assumption that no extractor is able to recover x. We point out that finding an obfuscation of f_x may not be efficient. However, the obfuscation itself is efficient because A is.

Amplifying knowledge extraction. Theorem 5.2.1 is not entirely satisfactory because extraction is guaranteed to occur only noticeably often (contrast this with the notions of Chapter 4 where extraction is required to succeed except with noticeable error). So, we address the issue of amplifying extraction. We show how to do so under a necessary (for the class of injective functions) and sufficient assumption on the function. Specifically, we assume what we call "weak verification". Weak verification is a notion introduced to show that some form of verification is necessary and sufficient for knowledge amplification. Moreover, it is implied by common verification notions such as public verification for probabilistic functions [Can97]. Informally, weak verification means for any adversary A that outputs images in the range of f, there is a corresponding verifier, V, which given some x and the private input of A, decides whether the output of A is a valid image of x under f. In other words, V has to decide whether there exists an r such that $f(x,r) = A(z, r_A)$, where z and r_A are the auxiliary information and random coins for A. Moreover, V is allowed to fail with some arbitrary small, yet noticeable probability. We use the terms "extraction (respectively, verification) with vanishing but noticeable error" and "extraction (respectively, verification) with arbitrary small but noticeable error" to mean that for every polynomial, p, there is an extractor (respectively, verifier) that fails no more than $\frac{1}{p}$ fraction of the time. The corresponding theorem can be stated in words as follows.

Theorem 5.2.3: Every weakly-verifiable family of probabilistic functions is either weakly obfuscatable or extractable with vanishing but noticeable error. Moreover, if an injective family of functions is extractable with vanishing but noticeable error, then it is weakly verifiable.

At a very high level, the proof of Theorem 5.2.3 uses a variant of Impagliazzo's hard-core lemma [Imp95] to amplify weak extraction to extraction with vanishing but noticeable error. Informally, we use the lemma to construct a family, $\mathbb{U}$, of machines that take the input of A and attempt to extract a preimage, x, from it. This family has

the property that when all its members fail, no machine can succeed *noticeably*. We then construct a family of distributions on the input of A, one distribution for each input length n, such that any member of $\mathbb{U}$ succeeds only negligibly often (as n increases). Consequently, if $\mathbb{U}$ is not a family of extractors with vanishing but noticeable error, then the distributions just mentioned have a noticeable weight in proportion to the original one. Using Theorem 5.2.1 on A and the new distributions implies the existence of an extractor with noticeable success. However, this contradicts the amplification lemma.

Interactively-extractable POW functions. An important corollary to Theorem 5.2.3 is that every POW function with auxiliary information is interactively extractable (see Corollary 5.2.3 for a more formal presentation). This supersedes the corresponding transformation of Chapter 4 from POW functions with auxiliary information to extractable POW functions. Moreover, this result is more efficient in that the challenger needs to send a single challenge instead of n.

Towards negligible error. We can obtain negligible failure probability if we relax the notion of extraction so that it applies only to "reliably-consistent adversaries". Intuitively, an adversary is reliably consistent if its consistency is noticeable. In other words, disregarding input on which the adversary is consistent only negligibly often, there is a fixed polynomial, p, such that $\frac{1}{p}$ is a lower bound on the probability of consistency (here, the probability is taken over the random challenge). The corresponding theorem can be stated as follows:

Theorem 5.2.5: Every weakly-verifiable family of probabilistic functions is either weakly obfuscatable or extractable with negligible error for adversaries that are reliably consistent.

Moreover, if an efficiently computable and verifiable family of functions is extractable with negligible error, then every corresponding adversary is reliably consistent.

The proof this theorem is very similar to the previous proof but it uses a stronger amplification lemma in the uniform model. Informally, the lemma states that there is a family, $\mathbb{U}$, of polynomial-time machines such that no machine can succeed in inverting a function where all members of $\mathbb{U}$ fail. (Contrast this lemma with the previous one, where the guarantee is that no machine can succeed *noticeably* where $\mathbb{U}$ fails.)

Uniform versus nonuniform extractors. We highlight that Theorems 5.2.1 and

5.2.3 deal with nonuniform extractors while Theorem 5.2.5 uses uniform extractors. Obviously, increasing the capabilities of extractors make them more powerful and more likely to recover preimages. In this case, giving extractors nonuniform capabilities is very beneficial in at least one case. Specifically, a nonuniform extractor that overwhelmingly fails in recovering a preimage implies that the distribution on the input domain is well-spread. The same statement does not immediately follow for uniform extractors. Consequently, functions are more likely to be extractable by nonuniform machines than uniform machines because functions are less likely to be weakly obfuscatable against *some well-spread distribution than against some distribution.* However, the nonuniform results do not follow through all the way to negligible error because negligible error use, in an essential way, properties of uniform machines. We refer the reader to Section 5.2.3 for a detailed presentation.

5.1.1.2 Noninteractive Extraction

Results similar to those for interactive extraction hold in this case. However, they are weaker in the sense that functions seem to be more likely to satisfy a weaker notion of obfuscation. Informally, the obfuscated program receives a function description, k, as input and outputs $f_k(x)$ for some x hidden in the program that may depend on k. Moreover, it is hard to recover x from the obfuscated code. The results and proofs are similar. Two issues are worth highlighting. First, the function is not fixed in advance. Rather, it is sampled from a well-spread distribution and given to the adversary. Second, a corollary to these results states that injective functions that are extractable with vanishing but noticeable error are extractable with negligible error.

5.1.2 Organization

We present the results for interactive extraction in Section 5.2 and for noninteractive extraction in Section 5.3.

5.2 Interactive Extraction versus Obfuscation

We present the three theorems mentioned in the introduction concerning the connection between obfuscation and interactive extraction with different extraction rates. Recall, the

first theorem says that every function is either weakly extractable or weakly obfuscatable. The second theorem builds on the first one to imply that every weakly verifiable function is either weakly obfuscatable or extractable with vanishing but noticeable error. The final theorem states that negligible-error extraction can be achieved if and only if certain conditions on the adversary are met. These conditions, termed “reliable consistency" in the introduction, are discussed and formalized in Section 5.2.3.

5.2.1 Weak Extraction

The statement that any function is either extractable or obfuscatable is to some degree intuitive. After all, these two notions are complementary in some way. For instance, suppose there is an obfuscated program that hides a license key inside it and is able to compute a new hash of the key. If we look at such a program from an extractability point of view, this means that there is a machine that simulates this program and computes the functionality mentioned above. Moreover, no extractor can recover the license key by the assumption that the obfuscated program hides it. Going in the reverse direction, it seems intuitive that the existence of an extractor for every adversary implies the absence of an obfuscation of such a functionality.

In the next theorem, we formalize and show that the intuition mentioned in the previous paragraph is sound. In more detail, statement 1 of this theorem (the obfuscation clause) states that there is a well-spread distribution, $\mathbb{X}$, on the input (think of this as the license key of the previous example) and an obfuscator, G_n, that takes a license key, x, and produces an obfuscated program, $g(x)$. In turn, $g(x)$ takes an input r and produces a new image of x using r as random coins for the function, i.e., $g(x)(r) = f(x, r)$. Moreover, $g(x)$ is required to be one-way in x but not required to succeed in computing this functionality more than noticeably often. In the theorem, we use the terminology $g(x)(\perp)$ to refer to a fixed hash of x available in the clear in the obfuscated program. On the other hand, statement 2 (the extraction clause) states that for any adversary, A, with any distribution on its input, z, r_A (z is auxiliary information and r_A is the random coins for A), that is consistent in the 3-round game discussed in Chapter 4, there is a corresponding extractor that recovers a preimage. In more detail, A is supposed to produce, with noticeable success, an image, y_0 in the first round and then again y_1 in the third round, such that there is a preimage common to both y_0 and y_1. Moreover,

the extractor is supposed to succeed only noticeably often.

Theorem 5.2.1. *Let $\mathbb{F} = \{f_n\}_{n\in\mathbb{N}}$ be any probabilistic family of functions and $\mathbb{R} = \{R_n\}_{n\in\mathbb{N}}$ be any distribution on the randomness domain of $\mathbb{F}$. Then, exactly one of the following two statements should hold:*

1. *There is a well-spread distribution $\mathbb{X}$ on the input domain of $\mathbb{F}$, a probabilistic function, $\mathbb{G} = \{G_n\}$ such that for any nonuniform polynomial-time machine, A:*
   (Obfuscation)

$$Pr[x \leftarrow X_n,\ g(x) \leftarrow G_n(x),\ x' = A(g(x)) : \exists r',\ g(x)(\perp) = f_n(x', r')] \leq \mu(n).$$

   (Functionality)

$$Pr[x \leftarrow X_n,\ g(x) \leftarrow G_n(x),\ r \leftarrow R_n : \exists r',\ g(x)(r) = f_n(x, r)\ \textit{and}$$

$$g(x)(\perp) = f_n(x, r')],$$

   is nonnegligible in n. Moreover, $g(x)(r)$ is efficiently computable, for any r.

2. *For any probabilistic polynomial-time machine, A, any infinite subset of security parameters, $\mathbb{N}'$, any distribution, $\mathbb{ZR} = \{ZR_n\}_{n\in\mathbb{N}'}$, on auxiliary information and the private input of A, if:*
   (Consistency)

$$Pr[(z, r_A) \leftarrow ZR_n,\ r_1 \leftarrow R_n,\ (y_0, s) = A(z, r_A),\ y_1 = A(s, r_1) :$$

$$\exists x',\ r_0,\ y_0 = f_n(x', r_0))\ \textit{and}\ y_1 = f_n(x', r_1))], \tag{5.1}$$

   is nonnegligible in n, then there exists a nonuniform polynomial-time machine, $\mathbb{K}$, such that:
   (Extraction)

$$Pr[(z, r_A) \leftarrow ZR_n,\ (y_0, s) = A(z, r_A),\ x = \mathcal{K}(z, r_A) : \exists r_0,\ y_0 = f_n(x, r_0)], \tag{5.2}$$

   is nonnegligible in n.

We emphasize that the previous theorem holds for any function. That is, it does not assume anything about the function, not even that it is efficiently computable. At a high level, the proof proceeds as follows. If f is not extractable, we take an adversary that violates this property and construct from it a distribution on the input of f (for clarity, refer to this as the license distribution) and an obfuscation on this distribution such that the obfuscation hides the license but is able to compute new images of it. In more detail, the license distribution is the distribution induced by A on preimages of its consistent output. For instance, if A *always* outputs $f_n(0, r_0)$ in the first round and $f_n(0, r_1)$ in the third round (in this case there is a straightforward extractor), then the induced distribution always samples 0. Moreover, the corresponding obfuscation is simply the input of A that causes A to output valid images of the license. Observe that the license distribution is well-spread because otherwise a nonuniform extractor can invert with noticeable probability. Therefore, using this license distribution with the corresponding obfuscation, statement 1 follows from the negation of statement 2. The other direction is easier to see and has been referred to in the first paragraph of this section. The formal proof follows.

Proof. ($\Longrightarrow$)

First, we show that if statement 2 does not hold, statement 1 should be true. Specifically, we construct, given that statement 2 does not hold, a well-spread distribution, $\mathbb{X}$, on the input domain and a corresponding function, G, that is "uninvertible" (as in the first requirement of statement 1) with respect to $\mathbb{X}$. Moreover, G helps in computing points in the range of F (as in the second requirement of statement 1). Putting these pieces together implies that statement 1 is true.

Formally, suppose that statement 2 does not hold. Then, there exists a PPT, A, an infinite set of security parameters, $\mathbb{N}'$, a distribution over auxiliary information and A's private input, $\mathbb{ZR}$, a polynomial, p_A, and an infinite subset of security parameters, $\mathbb{N}'' \subseteq \mathbb{N}'$ such that for all $n \in \mathbb{N}''$:

$$Pr[(z, r_A) \leftarrow ZR_n,\ r_1 \leftarrow R_n,\ (y_0, s) = A(z, r_A),\ y_1 = A(s, r_1) :$$

$$\exists x', r_0,\ y_0 = f_n(x', r_0) \text{ and } y_1 = f_n(x', r_1)] \geq \frac{1}{p_A(n)}, \tag{5.3}$$

and for any nonuniform polynomial-time machine, $\mathcal{K}$, and sufficiently large $n \in \mathbb{N}'$:

$$Pr[(z, r_A) \leftarrow ZR_n,\ x = \mathcal{K}(z, r_A) : \exists r_0,\ y_0 = f_n(x, r_0)] \leq \mu(n). \tag{5.4}$$

Eq. 5.4 has two major consequences. First, since all machines essentially fail in inverting y_0, then the distribution on the input induced by y_0 must be well-spread; otherwise the machine that receives the most frequent input as an advice string and outputs it yields a nonnegligible probability. Denote by $\mathbb{X}$ this distribution. Second, if we consider z, r_A as a function of x, then this function is univertible. Denote this function by $\mathbb{G}$. In the next two paragraphs, we present $\mathbb{X}$ and $\mathbb{G}$ in more detail.

Construction of $\mathbb{X}$.

Define the distribution $\mathbb{X} = \{X_n\}_{n \in \mathbb{N}''}$ as follows. For any a:

$$Pr[x \leftarrow X_n : x = a] =$$

$$\frac{1}{T_{X_n}} Pr[(z, r_A) \leftarrow ZR_n,\ r_1 \leftarrow R_n,\ (y_0, s) = A(z, r_A),\ y_1 = A(s, r_1) :$$

$$\exists r_0,\ y_0 = f_n(a, r_0) \text{ and } y_1 = f_n(a, r_1)],$$

where T_{X_n} is a normalizing factor, i.e.,

$$T_{X_n} = Pr[(z, r_A) \leftarrow ZR_n,\ r_1 \leftarrow R_n,\ (y_0, s) = A(z, r_A),\ y_1 = A(s, r_1) :$$

$$\exists x, r_0,\ y_0 = f_n(x, r_0) \text{ and } y_1 = f_n(x, r_1)]$$

By Eq. 5.3, $T_{X_n} > 0$ for all $n \in \mathbb{N}''$. Dividing by T_{X_n} ensures that $\mathbb{X}$ is a well-defined distribution.

Now, we show that $\mathbb{X}$ is well-spread. Suppose, for the purpose of contradiction, that it is not. Then, there is an element in the input domain that is sampled with a nonnegligible probability. Formally, for infinitely many n, there exists a polynomial $p_{\mathbb{X}}$ such that:

$$Pr[x \leftarrow X_n : x = \operatorname*{argmax}_a Pr[x' \leftarrow X_n : x' = a]] \geq \frac{1}{p_{\mathbb{X}}(n)}$$

Let $\mathcal{K}$ be a nonuniform machine that receives as an advice string $a_{max} = \operatorname*{argmax}_a Pr[x' \leftarrow$

$X_n : x' = a]$ and simply outputs it. We have for infinitely many n:

$$Pr[(z, r_A) \leftarrow ZR_n,\ x = \mathcal{K}(z, r_A) : \exists r_0,\ y_0 = f_n(x, r_0)] \geq \frac{1}{p_{\mathbb{X}}}.$$

A contradiction with Eq. 5.4. So, $\mathbb{X}$ must be well-spread.

Note that it is not clear how to efficiently sample an element from $\mathbb{X}$. However, if it is easy to sample from ZR_n, we can sample an image (under $\mathbb{F}$) of an element from X_n by choosing (z, r_A) from ZR_n and running A on z, r_A to get y_0.

Construction of $\mathbb{G}$. Let $\mathbb{G} = \{G_n\}_{n \in \mathbb{N}''}$ be a probabilistic function defined as follows. For any x and any b:

$$Pr[g(x) \leftarrow G_n(x) : g(x) = b] =$$

$$\begin{cases} 1 & \text{if } T_x = 0 \text{ and } b = \perp \\ 0 & \text{if } T_x = 0 \text{ and } b \neq \perp \\ \frac{1}{T_x} Pr[(z, r_A) \leftarrow ZR_n\ r_1 \leftarrow R_n,\ (y_0, s) = A(z, r_A), y_1 = A(s, r_1) : & \text{otherwise} \\ (z, r_A) = b \text{ and } \exists r_0,\ y_0 = f_n(x, r_0) \text{ and } y_1 = f_n(x, r_1)] & \end{cases} \tag{5.5}$$

Here again, T_x is a normalizing factor. It is the probability that A outputs valid images of x under $\mathbb{F}$. Formally, $T_x = Pr[(z, r_A) \leftarrow ZR_n,\ r_1 \leftarrow R_n,\ (y_0, s) = A(z, r_A), y_1 = A(s, r_1) : \exists r_0,\ y_0 = f_n(x, r_0) \text{ and } y_1 = f_n(x, r_1)]$.

We append to $g(x)$ some processing code that takes an input, r, and outputs an image of (x, r) under f_n, as computed by A on input z, r_A, r. Specifically,

$$g(x)(r) = \begin{cases} \perp & \text{if } g(x) = (z, r_A) = \perp \\ A(z, r_A) & \text{if } r = \perp \\ A(z, r_A, r) & \text{otherwise} \end{cases} \tag{5.6}$$

It is not clear how to efficiently compute $G_n(x)$ in general. However, $g(x)(r)$ is efficiently computable for any r.

We show that $\mathbb{G}$ is hard to invert by nonuniform polynomial-time machines. Observe that the distribution induced by $\mathbb{X}$ and $\mathbb{G}$ on (z, r_A) is the same as that of $\mathbb{ZR}$ *restricted* to those (z, r_A) which when we run A on them, A outputs valid images, y_0 and y_1. Formally,

Remark 5.2.1. *For any $n \in \mathbb{N}''$ and any $(z, r_A) \neq \perp$:*

$$Pr[x \leftarrow X_n,\ g(x) \leftarrow G_n : g(x) = (z, r_A)]$$

$$= Pr[(z', r'_A) \leftarrow ZR_n,\ r_1 \leftarrow R_n,\ (y_0, s) \leftarrow A(z', r'_A),\ y_1 = A(s, r_1) :$$

$$\exists x,\ r_0,\ y_0 = f_n(x, r_0) \text{ and } y_1 = f_n(x, r_1) \text{ and } (z', r'_A) = (z, r_A)]. \tag{5.7}$$

Eq. 5.7 follows from the constructions of $\mathbb{X}$ and $\mathbb{G}$, and from the observation that for any point, x, in the support of $\mathbb{X}$, $g(x)$ is never equal to $\perp$.

Thus, for any nonuniform polynomial-time machine, $\mathcal{K}$ and sufficiently large $n \in \mathbb{N}''$:

$$Pr[x \leftarrow X_n,\ g(x) \leftarrow G_n(x),\ x' = \mathcal{K}(g(x)) : \exists r_0,\ g(x)(\perp) = f_n(x', r_0)]$$

$$= Pr[(z, r_A) \leftarrow ZR_n,\ r_1 \leftarrow R_n,\ (y_0, s) \leftarrow A(z', r'_A),\ y_1 = A(s, r_1),\ x = \mathcal{K}(z, r_A) :$$

$$\exists r_0,\ y_0 = f_n(x, r_0) \text{ and } \exists x', r'_0,\ y_0 = f_n(x', r'_0) \text{ and } y_1 = f_n(x', r_1)]$$

$$\leq \mu(n), \tag{5.8}$$

where Eq. 5.8 follows from Eq. 5.3 and 5.4 and Remark 5.2.1.

Moreover, for any $n \in \mathbb{N}''$:

$$Pr[x \leftarrow X_n,\ g(x) \leftarrow G_n(x),\ r \leftarrow R_n :\ \exists r',\ g(x)(r) = f_n(x, r) \text{ and } g(x)(\perp) = f_n(x, r')]$$

$$= Pr[(z, r_A) \leftarrow ZR_n,\ r_1 \leftarrow R_n,\ (y_0, s) = A(z, r_A),\ y_1 = A(s, r_1) :$$

$$(\exists x',\ r_0,\ y_0 = f_n(x', r_0) \text{ and } y_1 = f_n(x', r_1))] \tag{5.9}$$

$$\geq \frac{1}{p_A(n)} \tag{5.10}$$

Eq. 5.9 and Eq. 5.10 hold by Remark 5.2.1 and Eq. 5.3, respectively. Eq. 5.8 and 5.10 imply statement 1. ($\Longleftarrow$)

Proving the reverse direction (if statement 1 holds, statement 2 should not) is easier. Let $\mathbb{X}$ and $\mathbb{G}$ be a pair of a well-spread distribution and probabilistic function satisfying the conditions of statement 1. Let $\mathbb{ZR}$ be the distribution induced by $\mathbb{X}$ on the range of

$\mathbb{G}$. Specifically, for any a:

$$Pr[z \leftarrow ZR_n : z = a] = Pr[x \leftarrow X_n,\ g(x) \leftarrow G_n(x) : g(x) = a].$$

Moreover, let A be an adversary that on input $g(x)$, outputs $g(x)(\perp)$, and on input $(g(x), r)$, outputs $g(x)(r)$. By the efficiency of $g(x)$, A is a PPT machine. From statement 1, it follows that $\mathbb{ZR}$ and A violate statement 2. □

Corollary 5.2.1. *Any deterministic one-way function is not even weakly extractable. That is, any deterministic one-way function satisfies statement* 1 *of Theorem 5.2.1. Moreover, this remains true if the function is not efficiently computable.*

Proof. Let f be any deterministic one-way function. Convert it into a probabilistic function that ignores the random coins, i.e. $F(x, r) = f(x)$.

Let A be a deterministic machine that receives $f(U_n)$ as auxiliary information and outputs it (as y_0 and then as a response for any challenge). Thus, A answers consistently with probability 1. On the other hand, the one-wayness property rules out the existence of a nonuniform polynomial-time machine that computes, with noticeable success, a preimage of the output of A. Consequently, f does not satisfy statement 2 of Theorem 5.2.1.

□

5.2.1.1 In the Uniform Setting

Theorem 5.2.1 was stated with respect to nonuniform extractors. In particular, the adversary, A, of statement 1 as well as the extractor, $\mathcal{K}$, of Statement 2 are nonuniform. Essentially, $\mathcal{K}$ is nonuniform to show that the distribution, $\mathbb{X}$, induced by $A(ZR_n)$ on the input domain is well-spread. And then this nonuniformity is passed on to statement 1 by negating statement 2.

If we consider a uniform extractor, $\mathcal{K}$, then statement 1 asserts the existence of a (not necessarily well-spread) distribution on which $\mathbb{G}$ is uninvertible by uniform polynomial-time machine. This version seems weaker than the first one. Thus, the negation of this statement (and consequently, the existence of statement 2) seems harder to achieve. We adopt the nonuniform version because statement 2 is one of the primary objectives of Theorem 5.2.1. On the other hand, we show in Section 5.2.3, how to extract with

negligible error in the uniform setting only. That result depends on the uniform version of Theorem 5.2.1, which we present here.

Theorem 5.2.2. *Let $\mathbb{F} = \{f_n\}_{n\in\mathbb{N}}$ be any probabilistic family of functions and $\mathbb{R} = \{R_n\}_{n\in\mathbb{N}}$ be any distribution on the randomness domain of $\mathbb{F}$. Then, exactly one of the following two statements should hold:*

1. *There is a distribution $\mathbb{X}$ on the input domain of $\mathbb{F}$, a probabilistic function, $\mathbb{G} = \{G_n\}$ such that for any deterministic polynomial-time machine, A:*
   (Obfuscation)

$$Pr[x \leftarrow X_n,\ g(x) \leftarrow G_n(x),\ x' = A(g(x)) : \exists r',\ g(x)(\perp) = f_n(x', r')] \leq \mu(n).$$

   (Functionality)

$$Pr[x \leftarrow X_n,\ g(x) \leftarrow G_n(x),\ r \leftarrow R_n : \exists r',\ g(x)(r) = f_n(x, r)\ \textit{and}$$

$$g(x)(\perp) = f_n(x, r')],$$

   is nonnegligible in n. Moreover, $g(x)(r)$ is efficiently computable, for any r.

2. *For any probabilistic polynomial-time machine, A, any infinite subset of security parameters, $\mathbb{N}'$, any distribution, $\mathbb{ZR} = \{ZR_n\}_{n\in\mathbb{N}'}$, on auxiliary information and the private input of A, if:*
   (Consistency)

$$Pr[(z, r_A) \leftarrow ZR_n,\ r_1 \leftarrow R_n,\ (y_0, s) = A(z, r_A),\ y_1 = A(s, r_1) :$$

$$\exists x',\ r_0,\ y_0 = f_n(x', r_0))\ \textit{and}\ y_1 = f_n(x', r_1))], \tag{5.11}$$

   is nonnegligible in n, then there exists a deterministic polynomial-time machine, $\mathbb{K}$, such that:
   (Extraction)

$$Pr[(z, r_A) \leftarrow ZR_n,\ (y_0, s) = A(z, r_A),\ x = \mathcal{K}(z, r_A) : \exists r_0,\ y_0 = f_n(x, r_0)], \tag{5.12}$$

   is nonnegligible in n.

The proof of this theorem is very similar to that of Theorem 5.2.1 and is not presented here.

5.2.2 Amplifying Extraction

Theorem 5.2.1 states that each function has a weakly extractable or weakly obfuscatable property. Next, we investigate conditions that allow for amplifying knowledge extraction. In particular, the goal in this section is to reach a vanishing but noticeable extraction error. Recall from the introduction, this term means that for every polynomial, p, there is an extractor that may depend on p and fails at most $\frac{1}{p}$ of the time. In Section 5.2.3, we address extraction with negligible error.

Not surprisingly, functions that admit such a property require more than the negation of statement 1 of Theorem 5.2.1. Recall that Theorem 5.2.1 holds for any function, in particular, not efficiently-computable functions. However, to reduce extraction error, efficient verification is needed. For the purpose of amplifying extraction, common notions of verification (e.g., Definition 2.5.1) are sufficient. However, a weaker but contrived form of verification is also sufficient, and, in the case of injective functions (i.e., for all y, there is no more than one x such that $y = f_n(x, r)$ for some r), is also necessary. Thus, we use this notion in the following theorem for the purpose of achieving a characterization instead of an implication. Informally, weak verification means there is a verifier tailored for every adversary, A. It receives x and the input of A and determines whether the output of A is a valid image of x. Moreover, the verifier is allowed to fail, when A is consistent, with noticeable probability.

Definition 5.2.1 (Weak Verification). *A function family , $\mathbb{F} = \{f_n\}_{n\in\mathbb{N}}$, satisfies weak verification if for every PPT, A (with input z, r_A), any distribution, $\mathbb{ZR} = \{ZR_n\}_{n\in\mathbb{N}'}$, on auxiliary information and the private input of A, and any polynomial p, there exists a nonuniform polynomial-time machine, $V_{A,ZR,p}$, such that for sufficiently large $n \in \mathbb{N}'$:*

$$Pr[(z, r_A) \leftarrow ZR_n,\ r_1 \leftarrow R_n,\ (y_0, s) = A(z, r_A),\ y_1 = A(s, r_1) :$$

$$(\exists x, r_0,\ V_{A,ZR,p}(x, z, r_A) \neq 1 \text{ and } f_n(x, r_0) = y_0$$

$$\text{or } \exists x, V_{A,ZR,p}(x, z, r_A) = 1 \text{ and } \forall r_0, f_n(x, r_0) \neq y_0)$$

$$\textit{and } (\exists x, r_0,\ f_n(x, r_0) = y_0 \textit{ and } f_n(x, r_1) = y_1)] < \frac{1}{p(n)}.$$

Theorem 5.2.3. *Let $\mathbb{F} = \{f_n\}_{n\in\mathbb{N}}$ be any probabilistic function family that is weakly extractable (satisfies statement 2 of Theorem 5.2.1). If $\mathbb{F}$ is weakly verifiable (as in Definition 5.2.1), then for any PPT A, any distribution, $\mathbb{ZR} = \{ZR_n\}_{n\in\mathbb{N}'}$, on auxiliary information and the private input of A, there exists a family of nonuniform polynomial-time machines, $\mathbb{U} = \{U_i\}_{i\in\mathbb{N}}$ such that for any polynomial p, there is an index i_p where for all $i \geq i_p$ and sufficiently large $n \in \mathbb{N}'$:*

$$Pr[(z, r_A) \leftarrow ZR_n,\ r_1 \leftarrow R_n,\ (y_0, s) = A(z, r_A),\ y_1 = A(s, r_1),\ x = U_i(z, r_A) :$$

$$(\exists r_0, f_n(x, r_0) = y_0 \textit{ or } (\forall x',\ (\forall r_0,\ y_0 \neq f_n(x', r_0)) \textit{ or } y_1 \neq f_n(x', r_1))] > 1 - \frac{1}{p(n)}. \tag{5.13}$$

Moreover, this implication is an equivalence for injective functions.

The proof uses, in an essential way, an amplification lemma similar to Impagliazzo's hard-core lemma [Imp95]. At a very high level, this lemma asserts the existence of a family of machines, $\mathbb{U}$, such that "no machine can succeed noticeably where all of these machines fail". Using this lemma, we then claim that for every polynomial, p, there is a member $U_{i_p} \in \mathbb{U}$ that fails in extracting a preimage with a probability at most $\frac{1}{p}$. If this were not to be the case, then this means there is some polynomial p, where every machine in $\mathbb{U}$ fails with probability at least $\frac{1}{p}$. This implies that there is a noticeable fraction of the domain where A is consistent yet all members of $\mathbb{U}$ fail. Lets restrict the distribution on the input of A to those on which such an event occurs. We then apply Theorem 5.2.1, in particular, statement 2, to obtain an extractor with noticeable success contradicting the lemma.

Proof. ($\Longrightarrow$)

The proof proceeds as follows. First, we present Lemma 5.2.1 that shows how to construct "strong" extractors from "weak" ones. Then, combining Lemma 5.2.1 with statement 2 of Theorem 5.2.1 yields Eq. 5.13.

In more detail, Lemma 5.2.1 says that there is a family of strong extractors with the property that if all members of this family fail in extracting preimages then so would all polynomial-time machines. We show that this family must indeed has arbitrary small

error because otherwise statement 2 implies the existence of an extractor that succeeds noticeably *where this family fails* contradicting the lemma.

Before we present the lemma formally, we clarify that it addresses function inversion in general and as such can be viewed as a version of Impagliazzo's hard-core lemma [Imp95]. In more detail, the function family, $\mathbb{F}$, mentioned in this lemma can be any function family and thus does not have to be associated with the function in the theorem. Later on, we prove the theorem by using this lemma on a function family related to the one in the statement of the theorem. Moreover, the lemma requires a distribution, $\mathbb{Y}$, on the output domain of $\mathbb{F}$ with a corresponding family of nonuniform deterministic polynomial-time weak verifiers, $V_{\mathbb{Y}} = \{V_{Y,n^i}\}_{i\in\mathbb{N}}$, for the support of $\mathbb{Y}$. Formally, for sufficiently large n: $Pr[y \leftarrow Y_n : \exists x, r,\ V_{Y,n^i}(x,y) \neq 1$ and $f_n(x,r) = y$ or $\exists x,\ V_{Y,n^i}(x,y) = 1$ and $\forall r, f_n(x,r) \neq y] < \frac{1}{n^i}$.

Lemma 5.2.1. *Let $\mathbb{F} = \{f_n\}_{n\in\mathbb{N}}$ be any probabilistic function family . Let $\mathbb{Y} = \{Y_n\}_{n\in\mathbb{N}}$ be any distribution on the output domain of $\mathbb{F}$ with a corresponding family of nonuniform polynomial-time weak verifiers, $V_{\mathbb{Y}} = \{V_{Y,n^i}\}_{i\in\mathbb{N}}$. Then there exists a family of nonuniform polynomial-time machines, $\mathbb{U} = \{U_i^j\}_{i\in\mathbb{N},j\in\mathbb{N}}$, satisfying three conditions:*

1. *For any nonuniform polynomial-time machine $\mathcal{K}$ with running time bounded by n^j, any index i, and sufficiently large n:*

$$Pr[y \leftarrow Y_n,\ x_1 = U_i^j(y),\ x_2 = \mathcal{K}(y) :\ \exists r, f_n(x_1,r) = y \text{ or } f_n(x_2,r) = y] -$$

$$Pr[y \leftarrow Y_n,\ x = U_i^j(y) :\ \exists r, f_n(x,r) = y] < \frac{1}{n^i}. \tag{5.14}$$

2. *If there is an infinite set of security parameters $\mathbb{N}'$, another distribution $\mathbb{Y}' = \{Y'_n\}_{n\in\mathbb{N}'}$, and a polynomial n^t, such that for all $n \in \mathbb{N}'$ and any a:*

$$Pr[y \leftarrow Y_n : y = a] \geq \frac{1}{n^t} Pr[y \leftarrow Y'_n : y = a], \tag{5.15}$$

and for all $i > 1$:

$$Pr[y \leftarrow Y'_n :\ \exists r f_n(U_i^i(y), r) = y] < \mu(n), \tag{5.16}$$

then for any nonuniform polynomial-time machine $\mathcal{K}$:

$$Pr[y \leftarrow Y'_n : \exists r f_n(\mathcal{K}(y), r) = y] < \mu(n). \tag{5.17}$$

3. *For any index $i > 1$ and any image y, if U_i^i succeeds in inverting y then so does U_{i+1}^{i+1}.*

Proof. This is a proof by construction. For any polynomial n^j and any i, we will define a nonuniform polynomial-time machine, U_i^j, that satisfies this lemma. Informally, U_i^j is given a set of nonuniform polynomial-time machines (with their corresponding advice strings) as an advice string. It simply simulates all of them on its input, y. If anyone succeeds in finding a preimage, it outputs that. Otherwise, it outputs $\perp$. The machines given in the advice string are chosen carefully to satisfy specific criterion. Specifically, each machine has a considerable success in finding a preimage, *where all others fail.* In more detail, the advice string contains all machines (running in time bounded by n^j) such that each one has *exclusively* at least a probability of $\frac{1}{n^i}$ in inverting $\mathbb{F}$. Due to their exclusive success probability, we can have, for any n, at most n^i such machines in the advice string. Therefore, the length of the advice string and consequently the running time of U_i^j is polynomially bounded. If any machine does not satisfy Eq. 5.14 then it should, by construction, be included in the advice string. Moreover, if it is in the advice string, then Eq. 5.14 must hold. The second property follows directly from Eq. 5.14; if there is any polynomial-time machine that succeeds with nonnegligible probability, then it contradicts Eq. 5.14 with respect to some *good* U_i^j. The third property is a technicality needed in the latter part of proof. It basically says if a good inverter fails then so do weaker ones.

Construction of U_i^j. Formally, denote by $a_i^j(n) = (b_i^j(n), c_i^j(n), a_i^V(n))$ the advice string of machine U_i^j for security parameter n, where $b_i^j(n)$ is an encoding of a set of nonuniform machines running in n^j time, and $c_i^j(n)$ consists of the advice strings for the corresponding machines in $b_i^j(n)$, and $a_i^V(n)$ is the advice string for the weak verifier, V_{Y,n^i} that fails with probability at most $\frac{1}{n^i}$. Initially, $b_i^j(n)$ (respectively, $c_i^j(n)$) is set to $b_{i-1}^{j-1}(n)$ (respectively, $c_{i-1}^{j-1}(n)$) with $a_1^j(n)$ and $a_i^1(n)$ set initially to ϵ. Then, any

nonuniform machine $\mathcal{K}$ with running time bounded by n^j is added to $b_i^j(n)$ if:

$$Pr[y \leftarrow Y_n,\ x = \mathcal{K}(y) : \exists r, f_n(x,r) = y \text{ and } \forall \mathcal{K}' \in b_i^j(n), x' = \mathcal{K}'(y) : \forall r', f_n(x',r') \neq y]$$

$$\geq \frac{1}{n^i}. \tag{5.18}$$

Finally, for every newly added machine, add its corresponding advice string for length n to $c_i^j(n)$. Since each machine in $b_i^j(n)$ exclusively contributes at least $\frac{1}{n^i}$ to the success probability, there can be at most n^i machines in $b_i^j(n)$ for any n. Moreover, since each machine has a running time bounded by n^j, it can be encoded as a string of length bounded by a polynomial q'. Likewise, the length of the advice string of each machine in $b_i^j(n)$ is bounded by n^j since no such machine can read more than n^j many symbols. Putting it all together, we have $|a_i^j(n)| = n^i(q'(n) + n^j) + |a_i^V(n)|$.

Now, U_i^j receives y as input and $a_i^j(n)$ as an advice string. It simulates all machines in $b_i^j(n)$ on y with their corresponding advice strings in $c_i^j(n)$. If any machine returns an x satisfying the condition $V_{Y,n^i}(x,y) = 1$, it returns x. Otherwise, it outputs $\perp$. Note that the running time of U_i^j is bounded by some fixed polynomial in $n^i n^j$.

Observe if for some $i > 1$, U_i^i succeeds in inverting any y then there is a machine in $b_i^i(n)$ that succeeds as well. By construction $b_i^i(n)$ is in $b_{i+1}^{i+1}(n)$. So, U_{i+1}^{i+1} also succeeds in inverting y (assuming $V_{\mathbb{Y},n^{i+1}}$ does no worse than $V_{\mathbb{Y},n^i}$), and property 3 holds.

Proof of Eq. 5.14. Suppose, for the purpose of contradiction, that there is a nonuniform machine, $\mathcal{K}$, running in time at most n^j, an index i, and some arbitrary large n such that:

$$Pr[y \leftarrow Y_n,\ x_1 = U_i^j(y),\ x_2 = \mathcal{K}(y) :\ \exists r, f_n(x_1,r) = y \text{ or } f_n(x_2,r) = y] -$$

$$Pr[y \leftarrow Y_n,\ x = U_i^j(y) :\ \exists r, f_n(x,r) = y] \geq \frac{1}{n^i}.$$

There are two cases. First, $\mathcal{K}$ is in $b_i^j(n)$. Then by construction, the difference above is at most the error of V_{Y,n^i} which is less than $\frac{1}{n^i}$. Second, $\mathcal{K}$ is not in $b_i^j(n)$. In this case, $\mathcal{K}$ satisfies Eq. 5.18. So, it should be in $b_i^j(n)$. In either case, there is a contradiction.

Proof of Property 2. We show that property 2 is true based on Eq. 5.14. Suppose,

for the purpose of contradiction, that Eq. 5.15 and Eq. 5.16 hold but Eq. 5.17 does not. Then, there exists a nonuniform machine, $\mathcal{K}$, with running time bounded by some polynomial, n^j, another polynomial n^i, such that for infinitely many $n \in \mathbb{N}'$:

$$Pr[y \leftarrow Y'_n : \exists r f_n(\mathcal{K}(y), r) = y] > \frac{1}{n^i}, \tag{5.19}$$

Let $t' = 2(max(i,j)+t)$. Then, we have for infinitely many $n \in \mathbb{N}'$:

$$Pr[y \leftarrow Y_n,\ x_1 = U_{t'}^{t'}(y),\ x_2 = \mathcal{K}(y) : \exists r f_n(x_1, r) = y \text{ or } f_n(x_2, r) = y] -$$

$$Pr[y \leftarrow Y_n,\ x = U_{t'}^{t'}(y) : \exists r f_n(x, r) = y]$$

$$= Pr[y \leftarrow Y_n,\ x_1 = U_{t'}^{t'}(y),\ x_2 = \mathcal{K}(y) : \forall r, f_n(x_1, r) \neq y \text{ and } \exists r, f_n(x_2, r) = y]$$

$$\geq \frac{1}{n^t} Pr[y \leftarrow Y'_n,\ x_1 = U_{t'}^{t'}(y),\ x_2 = \mathcal{K}(y) : \forall r, f_n(x_1, r) \neq y \text{ and } \exists r, f_n(x_2, r) = y] \tag{5.20}$$

$$\geq \frac{1}{n^t}(1 - \mu(n))(\frac{1}{n^i} - \mu(n)) \tag{5.21}$$

$$\geq \frac{1}{n^{t'}},$$

where Eq. 5.20 follows from Eq. 5.15, and Eq. 5.21 follows from Eq. 5.19 and Eq. 5.16. A contradiction with Eq. 5.14. □

Lemma 5.2.1 + statement 2 of Theorem 5.2.1 $\implies$ Eq. 5.13 of Theorem 5.2.3.

If statement 2 of Theorem 5.2.1 holds, then Lemma 5.2.1 implies Eq. 5.13. Again, this is a proof by contradiction. Suppose the inequality of Theorem 5.2.3 does not hold. Then, there is a weakly-extractable and weakly-verifiable probabilistic function, $\mathbb{F}$, a PPT A, an infinite set of security parameters, $\mathbb{N}'$, a distribution on the auxiliary information and A's private input, $\mathbb{ZR}^1$, and a polynomial p such that for any nonuniform polynomial-time machine, $\mathcal{K}$, there is an infinite subset of security parameters $\mathbb{N}_{\mathcal{K}} \subseteq \mathbb{N}'$ such that:

$$Pr[(z, r_A) \leftarrow ZR_n^1,\ r_1 \leftarrow R_n,\ (y_0, s) = A(z, r_A),\ y_1 = A(s, r_1), x = \mathcal{K}(z, r_A) :$$

$$(\forall r, f_n(x,r) \neq y_0 \text{ and } (\exists x',\ r_0,\ y_0 = f_n(x',r_0) \text{ and } y_1 = f_n(x',r_1))] \geq \frac{1}{p(n)}. \quad (5.22)$$

Let $\mathbb{ZR}^2 = \{ZR_n^2\}_{n\in\mathbb{N}'}$ be the restriction of $\mathbb{ZR}^1$ to those elements that cause A to output a consistent pair of images, y_0 and y_1. Formally, for any $n \in \mathbb{N}'$ and any (a,b):

$$Pr[(z,r_A) \leftarrow ZR_n^2 : (z,r_A) = (a,b)] =$$

$$\frac{1}{T_{ZR_n^2}} Pr[(z,r_A) \leftarrow ZR_n^1,\ r_1 \leftarrow R_n,\ (y_0,s) = A(z,r_A),\ y_1 = A(s,r_1) :$$

$$(z,r_A) = (a,b) \text{ and } \exists x', r_0,\ y_0 = f_n(x',r_0) \text{ and } y_1 = f_n(x',r_1)],$$

where $T_{ZR_n^2} = Pr[(z,r_A) \leftarrow ZR_n^1 :\ r_1 \leftarrow R_n,\ (y_0,s) = A(z,r_A),\ y_1 = A(s,r_1) : \exists x', r_0,\ y_0 = f_n(x',r_0) \text{ and } y_1 = f_n(x',r_1)]$. W.l.o.g., $\mathbb{N}'$ is restricted to security parameters for which A succeeds in answering consistently with nonzero probability. Consequently, the denominator of the previous fraction is nonzero for all $n \in \mathbb{N}'$.

Let $\mathbb{G} = \{G_n\}_{n\in\mathbb{N}'}$ be a probabilistic function defined as follows. For any $n \in \mathbb{N}'$ and (z,r_A):

$$Pr[g(x) \leftarrow G_n(x) : g(x) = z, r_A] =$$

$$\begin{cases} 1 & \text{if } T_x = 0 \text{ and } (z,r_A) = \perp \\ 0 & \text{if } T_x = 0 \text{ and } (z,r_A) \neq \perp \\ 0 & \text{if } \forall r_0,\ y_0 \neq f_n(x,r_0), \text{ where } (y_0,s) = A(z,r_A) \\ \frac{Pr[(z',r'_A) \leftarrow ZR_n^1 : (z',r'_A) = (z,r_A)]}{Pr[(z,r_A) \leftarrow ZR_n^1,\ (y_0,s) = A(z,r_A) : \exists r_0,\ y_0 = f_n(x,r_0)]} & \text{if } \exists r_0,\ y_0 = f_n(x,r_0), \text{ where } (y_0,s) = A(z,r_A) \end{cases} \quad (5.23)$$

Here, $T_x = Pr[(z,r_A) \leftarrow ZR_n^1,\ (y_0,s) = A(z,r_A) : \exists r_0,\ y_0 = f_n(x,r_0)]$. Note that $\mathbb{ZR}^2$ is a distribution on the range of $\mathbb{G}$ because no element, (z,r_A), can be in the support of $\mathbb{ZR}^2$ unless y_0 has a valid preimage under f_n. That is, $(y_0,s) = A(z,r_A)$ and $\exists x, r, f_n(x,r) = y_0$. This makes (z,r_A) one of the possible images of x under G_n.

Associate with the distribution, $\mathbb{ZR}^2$, the family of weaker verifiers, $\mathbb{V}_{\mathbb{ZR}^2} = \{V_{\mathbb{ZR}^2,n^i}\}_{n\in\mathbb{N}'}$, where $V_{\mathbb{ZR}^2,n^i} = V_{A,\mathbb{ZR}^2,n^i}$.

$\mathbb{V}_{\mathbb{ZR}^2}$ is a weak verifier for $\mathbb{ZR}^2$.

By construction, we have for sufficiently large n:

$$Pr[(z,r_A) \leftarrow ZR_n^2 : \exists x, r,\ V_{ZR^2,n^i}(x,z,r_A) \neq 1 \text{ and } g_n(x,r) = (z,r_A)$$

$$\text{or } \exists x,\ V_{ZR^2,n^i}(x,z,r_A) = 1 \text{ and } \forall r, g_n(x,r) \neq (z,r_A)]$$

$$= Pr[(z, r_A) \leftarrow ZR_n^1,\ r_1 \leftarrow R_n,\ (y_0, s) = A(z, r_A),\ y_1 = A(s, r_1) :$$

$$(\exists x', r_0,\ y_0 = f_n(x', r_0) \text{ and } y_1 = f_n(x', r_1)) \text{ and}$$

$$(\exists x, r,\ V_{A,n^i}(x, z, r_A) \neq 1 \text{ and } y_0 = f_n(x, r_0) \text{ or}$$

$$\exists x,\ V_{A,n^i}(x, z, r_A) = 1 \text{ and } \forall r, y_0 \neq f_n(x, r_0)] < \frac{1}{n^i},$$

where the last inequality follows from Definition 5.2.1.

Now, we use Lemma 5.2.1 with the parameters: $\mathbb{G}$ (as the weakly-verifiable function), $\mathbb{ZR}^2$ (as the distribution on the output domain of $\mathbb{G}$), and $\mathbb{V}_{\mathbb{ZR}^2}$ (as a weak verifier) to obtain the family, $\mathbb{U}$, as described in that lemma.

By the definition of $\mathbb{ZR}^2$ and then by Eq. 5.22, we have for any U_i^i, there is an infinite subset of security parameters $\mathbb{N}_{U_i^i} \subseteq \mathbb{N}'$ such that:

$$Pr[(z, r_A) \leftarrow ZR_n^2,\ (y_0, s) = A(z, r_A),\ x = U_i^i(z, r_A) : \forall r, f_n(x, r) \neq y_0]$$

$$= Pr[(z, r_A) \leftarrow ZR_n^1,\ r_1 \leftarrow R_n,\ (y_0, s) = A(z, r_A),\ y_1 = A(s, r_1), x = U_i^i(z, r_A) :$$

$$(\forall r, f_n(x, r) \neq y_0 \text{ and } (\exists x',\ r_0,\ y_0 = f_n(x', r_0) \text{ and } y_1 = f_n(x', r_1))] \geq \frac{1}{p(n)}. \quad (5.24)$$

We will use Eq. 5.24 to derive a special distribution on auxiliary information and private input of A. This distribution allows A to answer challenges consistently with nonnegligible probability while all U_i^i fail in finding a preimage with overwhelming probability. Given such a distribution, Lemma 5.2.1 implies that all polynomial-time machines succeed with negligible probability while statement 2 of Theorem 5.2.1 tells us that some machine succeeds with *nonnegligible* probability.

By property 3 of Lemma 5.2.1, we have $\mathbb{N}_{U_{i+1}^{i+1}} \subseteq \mathbb{N}_{U_i^i}$ for all i. Let $\mathbb{N}_\infty = \cup_{i=1}^\infty \{a_i\}$, where $a_i = \inf \mathbb{N}_{U_i^i} - \{a_1, ..., a_{i-1}\}$.

Remark 5.2.2. *Note that* $|\mathbb{N}_\infty| = \infty$ *and Eq. 5.24 holds for all* U_i^i *and all* $n \geq a_i$, *where* $n \in \mathbb{N}_\infty$.

Let $\mathbb{ZR}^3 = \{ZR_n^3\}_{n \in \mathbb{N}_\infty}$ be the restriction of $\mathbb{ZR}^2$ to those elements on which the family $\mathbb{U}$ fails. Formally, for any $n = a_i \in \mathbb{N}_\infty$, ZR_n^3 is defined as follows:

$$Pr[(z, r_A) \leftarrow ZR_n^3 : (z, r_A) = (a, b)]$$

$$= \frac{1}{T_{ZR_n^3}} Pr[(z, r_A) \leftarrow ZR_n^2,\ x = U_i^i(z, r_A),\ (y_0, s) = A(z, r_A) :$$

$$\forall r, f_n(x, r) \neq y_0 \text{ and } (z, r_A) = (a, b)],$$

where $T_{ZR_n^3} = Pr[(z, r_A) \leftarrow ZR_n^2,\ x = U_i^i(k, z, r_A),\ (y_0, s) = A(z, r_A) : \forall r, f_n(x, r) \neq y_0]$. By Remark 5.2.2 and Eq. 5.24, $T_{ZR_n^3} \geq \frac{1}{p(n)}$, for all $n \in \mathbb{N}_\infty$. Therefore, for any $n = a_i \in \mathbb{N}_\infty$ and any (a, b):

$$Pr[(z, r_A) \leftarrow ZR_n^2 : (z, r_A) = (a, b)]$$

$$\geq \begin{cases} \frac{1}{p(n)} Pr[(z, r_A) \leftarrow ZR_n^3 : (z, r_A) = (a, b)] = 0 & \text{if } \exists r, f_n(x, r) = y_0, \text{ where:} \\ & x = U_i^i(a, b) \text{ and } (y_0, s) = A(a, b) \\ \frac{1}{p(n)} Pr[(z, r_A) \leftarrow ZR_n^3 : (z, r_A) = (a, b)] & \text{otherwise} \end{cases} \tag{5.25}$$

Moreover, by property 3 of Lemma 5.2.1 and by definition of $\mathbb{ZR}^3$, we have for any i and all $n \in \mathbb{N}_\infty$, $n \geq a_i$:

$$Pr[(z, r_A) \leftarrow ZR_n^3,\ x = U_i^i(z, r_A),\ (y_0, s) = A(z, r_A) : \exists r, f_n(x, r) = y_0] = 0. \tag{5.26}$$

Eq. 5.25, 5.26, and Lemma 5.2.1 (in particular, the second property) imply that Eq. 5.17 should hold with respect to any nonuniform polynomial-time machine. Plugging in the correct parameters, we have for any nonuniform polynomial-time machine, $\mathcal{K}$, and $n \in \mathbb{N}_\infty$:

$$Pr[(z, r_A) \leftarrow ZR_n^3,\ (y_0, s) = A(z, r_A), x = \mathcal{K}(z, r_A) : \exists y_0, f_n(x, r_0) = y_0] < \mu(n). \tag{5.27}$$

On the other hand, for any $n = a_i \in \mathbb{N}_\infty$:

$$Pr[(z, r_A) \leftarrow ZR_n^3,\ r_1 \leftarrow R_n,\ (y_0, s) = A(z, r_A),\ y_1 = A(s, r_1) :$$

$$\exists x', r_0,\ y_0 = f_n(x', r_0) \text{ and } y_1 = f_n(x', r_1)]$$

$$= Pr[(z, r_A) \leftarrow ZR_n^1,\ r_1 \leftarrow R_n,\ (y_0, s) = A(z, r_A),\ y_1 = A(s, r_1),\ x = U_i^i(z, r_A) :$$

$$\exists x', r_0,\ y_0 = f_n(x', r_0) \text{ and } y_1 = f_n(x', r_1) \text{ and } \forall r, f_n((x, r) \neq y_0]$$

$$\geq \frac{1}{p(n)}. \tag{5.28}$$

The last inequality follows from Eq. 5.24.

Eq. 5.28 and statement 2 of Theorem 5.2.1 imply that there exists a nonuniform polynomial-time machine satisfying Eq. 5.12. This contradicts Eq. 5.27. Therefore, Eq. 5.13 holds.

($\Longleftarrow$)

The converse of the above result is true for injective functions. Specifically, any injective function that satisfies Eq. 5.13 is weakly verifiable. This is so because for such functions, an extractor can be easily transformed into a verifier. Formally, for any A (with input z, r_A), any distribution, $\mathbb{ZR}$, and any polynomial p, let $\mathcal{K}$ be the corresponding extractor satisfying Eq. 5.13. Let $V_{A,\mathbb{ZR},p}(x, z, r_A) = 1$ if and only if $x = \mathcal{K}(z, r_A)$. Note that if $\mathcal{K}$ succeeds in computing a preimage, then $V_{A,\mathbb{ZR},p}$ behaves correctly on z, r_A and for any x. Thus, $V_{A,\mathbb{ZR},p}$ fails no more than $\frac{1}{p(n)}$ of the time, for sufficiently large n. □

Corollary 5.2.2. *If $\mathbb{F} = \{f_n\}_{n\in\mathbb{N}}$ is a probabilistic function family that is efficiently computable, has public randomness, and satisfies statement 2 of Theorem 5.2.1, then $\mathbb{F}$ is extractable with arbitrary small, yet noticeable error (as in Eq. 5.13).*

Proof. Associate with $\mathbb{F}$ the following verifiable: $V_{\mathbb{F}}(x, y = (r, y')) = 1$ if and only if $f_n(x, r) = y$. It follows then that $\mathbb{F}$ is weakly verifiable (let $V_{A,ZR,p}(x, z, r_A) = V_{\mathbb{F}}(x, y_0)$, where $A(z, r_A) = y_0, s$). Apply Theorem 5.2.3 to get the result. □

The following corollary is one of the main applications of this result.

Corollary 5.2.3. *Every POW function with auxiliary information that is collision resistant and has public randomness is interactively-extractable with vanishing but noticeable error (as in Theorem 5.2.3).*

Proof. Let $\mathbb{H} = \{\{H_k\}_{k\in K_n}\}_{n\in\mathbb{N}}$ be a family ensemble of POW functions with auxiliary information. Pick any sequence of functions, $H = \{H_{k_n}\}_{n\in\mathbb{N}}$, where $H_{k_n} \in K_n$. We argue that H does not satisfy statement 1 of Theorem 5.2.1 because H is perfectly one-way with auxiliary information. Suppose, for the purpose of contradiction, that there is a well-spread distribution, $\mathbb{X}$ and a function $\mathbb{G}$, satisfying statement 1 of Theorem 5.2.1. By assumption, $\mathbb{G}$ is one-way and consequently can be used as auxiliary information to H. Let A be an adversary that receives $g(x), y$ as input, where y can be either $H_{k_n}(x, r)$

or $H_{k_n}(u,r)$ and u and r are uniform. By public randomness, r is in the input to A as part of $H_k(.,r)$. Now, A computes $g(x)(r)$ and outputs 1 if $y = g(x)(r)$. Otherwise, it outputs 1 with probability $\frac{1}{2}$. In the case where $y = H_{k_n}(x,r)$, A outputs 1 with probability noticeable better than $\frac{1}{2}$. However, if $y = H_{k_n}(u,r)$, then A outputs 1 with probability negligibly close to $\frac{1}{2}$ because of collision resistance. This contradicts perfect one-wayness. Thus, H satisfies statement 2 of Theorem 5.2.1. Using, in addition, the fact that H is efficiently computable and has public randomness, Corollary 5.2.2 implies that H satisfies Eq. 5.13. □

5.2.2.1 In the Uniform Model

The uniform version of Theorem 5.2.3 is very similar. The only difference is that both the verifier and extractor are deterministic polynomial-time machines. Moreover, the proof follows the same lines as that of Theorem 5.2.3. However, the amplification lemma is quite stronger in this setting. In fact, this is one of the reason that makes negligible extraction error possible in Section 5.2.3. In words, this lemma provides a family of extractors such that no machine can succeed *even negligibly* often where this family fails.

Definition 5.2.2 (Weak Verification (uniform)). *A function family , $\mathbb{F} = \{f_n\}_{n\in\mathbb{N}}$, satisfies weak verification if for every PPT, A (with input z, r_A), any distribution, $\mathbb{ZR} = \{ZR_n\}_{n\in\mathbb{N}'}$, on auxiliary information and the private input of A, and any polynomial p, there exists a deterministic polynomial-time machine, $V_{A,ZR,p}$, such that for sufficiently large $n \in \mathbb{N}'$:*

$$Pr[(z,r_A) \leftarrow ZR_n,\ r_1 \leftarrow R_n,\ (y_0,s) = A(z,r_A),\ y_1 = A(s,r_1) :$$

$$(\exists x, r_0,\ V_{A,ZR,p}(x,z,r_A) \neq 1 \text{ and } f_n(x,r_0) = y_0$$

$$\text{or } \exists x, V_{A,ZR,p}(x,z,r_A) = 1 \text{ and } \forall r_0, f_n(x,r_0) \neq y_0)$$

$$\text{and } (\exists x, r_0,\ f_n(x,r_0) = y_0 \text{ and } f_n(x,r_1) = y_1)] < \frac{1}{p(n)}.$$

Theorem 5.2.4. *Let $\mathbb{F} = \{f_n\}_{n\in\mathbb{N}}$ be any probabilistic function family that is weakly extractable (satisfies statement 2 of Theorem 5.2.2). If $\mathbb{F}$ is weakly verifiable (as in Definition 5.2.2), then for any PPT A, any distribution, $\mathbb{ZR} = \{ZR_n\}_{n\in\mathbb{N}'}$, on auxiliary*

information and the private input of A, there exists a family of deterministic polynomial-time machines, $\mathbb{U} = \{U_i\}_{i\in\mathbb{N}}$ such that for any polynomial p, there is an index i_p where for all $i \geq i_p$ and sufficiently large $n \in \mathbb{N}'$:

$$Pr[(z, r_A) \leftarrow ZR_n,\ r_1 \leftarrow R_n,\ (y_0, s) = A(z, r_A),\ y_1 = A(s, r_1),\ x = U_i(z, r_A) :$$

$$(\exists r_0, f_n(x, r_0) = y_0 \text{ or } (\forall x',\ (\forall r_0,\ y_0 \neq f_n(x', r_0)) \text{ or } y_1 \neq f_n(x', r_1))] > 1 - \frac{1}{p(n)}. \tag{5.29}$$

Moreover, this implication is an equivalence for injective functions.

The proof is very similar to that of Theorem 5.2.3 except that it uses the following amplification lemma instead of Lemma 5.2.1. Informally, this lemma provides a family of machines, $\mathbb{U}$, such that any machine can not succeed even negligibly where this family fails. At a high level, each $U_i \in \mathbb{U}$ contains the first i machines in an enumeration of uniform polynomial-time machines. This ensures that every polynomial-time machine is eventually included in this family.

Lemma 5.2.2. *Let $\mathbb{F} = \{f_n\}_{n\in\mathbb{N}}$ be any probabilistic function family . Let $\mathbb{Y} = \{Y_n\}_{n\in\mathbb{N}}$ be any distribution on the output domain of $\mathbb{F}$ with a corresponding family of weak verifiers, $V_{\mathbb{Y}} = \{V_{Y,n^i}\}_{i\in\mathbb{N}}$. Then there exists a family of uniform PT machines, $\mathbb{U} = \{U_i\}_{i\in\mathbb{N}}$, satisfying the following two conditions:*

1. *for any $i, j,\ i < j$ and all n:*

$$Pr[y \leftarrow Y_n,\ x = U_i(y) : \exists r,\ f_n(x, r) = y] \leq Pr[y \leftarrow Y_n,\ x = U_j(y) : \exists r,\ f_n(x, r) = y].$$

2. *For any distribution $\mathbb{Y}'$ with a corresponding family of weaker verifiers, any PT machine, $\mathcal{K}$, $\exists i$ such that for all $j \geq i$ and sufficiently large n:*

$$Pr[y \leftarrow Y'_n,\ x = \mathcal{K}(y) : \exists r,\ f_n(x, r) = y] \leq$$

$$Pr[y \leftarrow Y'_n,\ x = U_j(y) : \exists r,\ f_n(x, r) = y] + \frac{1}{n^j}.$$

Proof. Let $S = \{M_1^1, M_1^2, M_2^2, ...\}$ be an enumeration of deterministic polynomial-time machines. For instance, if $S' = \{M_1, M_2, ...\}$ is an enumeration of Turing machines, then

$M_i^j \in S$ corresponds to the i^{th} machine in S' with a timer that stops M_i after n^j steps. Let U_i be the machine that simulates every member of $S_i = \{M_1^1, M_1^2, ..., M_i^i\}$ on its input, y. Then, it runs V_{A,n^i} on the output of each machine and returns the one that V_{A,n^i} accepts (or $\perp$ if V_{A,n^i} does not accept any).

First condition. The first condition of the lemma follows immediately from the fact that for any i, j, $i < j$, U_j simulates all the machines that U_i does and also uses a more accurate verifier (assuming V_{A,n^i} does no better than V_{A,n^j}).

Second condition. Let $\mathcal{K}$ be any machine with polynomial running time. Then, by construction, there exists an i such that for all $j \geq i$, U_j simulates $\mathcal{K}$. Thus, when $\mathcal{K}$ succeeds then U_j succeeds as well, except when the verifier fails. By definition, the latter event happens with probability $\frac{1}{n^j}$. □

5.2.3 Towards Extraction with Negligible Error

The previous section underscores conditions that are necessary (at least for injective functions) and sufficient for extraction with vanishing but noticeable error. Here, we address the question of obtaining extraction with negligible error. As before, we show necessary and sufficient conditions to achieve this objective. However, unlike the previous results, the conditions are not on the function but rather on the adversary itself. Moreover, as we discuss later on, this result is in the uniform setting only.

Conditions for extraction with negligible error. As we mentioned in the introduction, extraction with negligible error requires "reliable consistency" on the behalf of the adversary. Informally, we show that negligible extraction error is possible for a particular adversary, A, if it can answer challenges consistently with probability bounded from below by the inverse of some fixed polynomial. Informally, it may be the case that A answers consistently with noticeable probability. Yet, depending on its input, the probability of its consistency (taken over the random coins of the challenger) can be arbitrary small though still noticeable. In such a scenario, extraction can not achieve negligible error because as answers are less likely to be consistent, extraction requires more effort and time to find a preimage. On the other hand, if for almost all of its input, A answers consistently with a probability bounded from below by an inverse polynomial, this bound can be translated into an upper bound on the running time of the extractor.

We elaborate on these conditions through a toy example. Suppose there is a function,

f and an adversary A with the following properties. A outputs a consistent pair (y_0, y_1) with probability $\frac{1}{n^i}$ for every element in the i^{th} $\frac{2^n}{n}$ fraction of the input domain of A. Here, the probability is taken over random coins sent by the challenger in round 2. Formally, we have for every n, and every $(z, r_A) \in [\frac{i2^n}{n}, \frac{(i+1)2^n}{n}]$:

$$Pr[r_1 \leftarrow R_n,\ (y_0, s) = A(z, r_A),\ y_1 = A(s, r_1) : \exists x, r_0,\ f_n(x, r_0) = y_0$$

$$\text{and } f_n(x, r_1) = y_1] = \frac{1}{n^i}.$$

Now, it may be the case that extraction depends on how successful A is in answering challenges. If this is so, then extraction is proportional to consistency. In other words, the more consistent A is, the less time extraction takes. In such a scenario, it turns out that overwhelming success requires super-polynomial time. In other words, noticeable extraction error is unavoidable.

In the previous example, we assume that A has a noticeable success in every fraction of the input domain. Also, we assume that A can not do any better. In other words, A can not amplify its success rate. However, there are cases where A can indeed amplify its success, e.g., A may provide wrong answers intentionally even though it can easily compute the correct ones. In such a scenario, extraction with negligible error is possible. As an example, consider an adversary, A, that provides wrong answers intentionally. A receives x as input, computes i such that $x \in [\frac{i2^n}{n}, \frac{(i+1)2^n}{n}]$, and gives the correct answer only if $r_1 \in [0, \frac{2^n}{n^i}]$. Even though A satisfies the previous condition, an extractor can easily recover x by reading it from the input. So, we need a meaningful way to separate the notion of "truthful" failure from "intentional" failure. In the next theorem, we capture the notion of intentional failure through the existence of another machine A' that behaves similarly to A, yet it amplifies its consistency.

Uniform Setting. The proof of Theorem 5.2.3 uses a diagonalization technique to show that no machine can succeed "substantially" where the family $\mathbb{U}$ fails. The diagonalization is over machines that succeeds noticeably over inputs of some length n. This technique works because this set of machines is enumerable. (Specifically, there are at most n machines that each succeeds exclusively with probability $\frac{1}{n}$ and so on.) However, this technique fails when we try to use it to achieve negligible error in polynomial time. Two factors seem to prevent this technique from working. First, the set of nonuniform

polynomial-time machines is not enumerable and so we can not diagonalize over this set (as we discuss later on, we use enumeration of uniform machines to prove this result in the uniform setting). Second, if we instead consider machines that succeed exclusively, as in the previous theorem, we need to take into account those that succeed with negligible probability, yet the probability is not "very negligible", say, $\frac{1}{n^{logn}}$. However, this causes $\mathbb{U}$ to be slightly super-polynomial. Consequently, the next theorem applies to the uniform setting only. It is based on Theorem 5.2.2 instead of Theorem 5.2.1.

Before we present the theorem we describe reliable consistency in more detail. Reliable consistency refers to a new machine, A', that replaces an adversary, A, with the purpose of undoing any intentional failure on behalf of A. The conditions on A' are as follows:

1. The output of A' is equivalent to A in the first round.
2. The consistency of A' is not any worse than that of A.
3. There is a fixed polynomial, $p_{A'}$, such that almost all inputs to A' cause it to be either consistent negligibly or with probability at least $\frac{1}{p_{A'}}$.

If there is such an A' then extraction with negligible extraction error is possible. Moreover, the converse is also true for efficiently computable and verifiable functions.

Theorem 5.2.5. *Let $\mathbb{F} = \{f_n\}_{n\in\mathbb{N}}$ be any probabilistic function family that satisfies statement 2 of Theorem 5.2.2 and is weakly verifiable (as in Definition 5.2.2).*

Let A be any PPT and $\mathbb{ZR} = \{ZR_n\}_{n\in\mathbb{N}'}$ be any distribution on auxiliary information and the private input of A. If there is another PPT, A', satisfying the following three conditions of reliable consistency:

1. *$A'(z, r_A) = A(z, r_A)$ for all z, r_A.*
2. $$Pr[(z, r_A) \leftarrow ZR_n,\ r_1 \leftarrow R_n,\ (y_0, s) = A'(z, r_A),\ y_1 = A'(s, r_1) :$$
$$\exists x',\ r_0,\ y_0 = f_n(x', r_0))\ \textit{and}\ y_1 = f_n(x', r_1))]$$
$$\geq Pr[(z, r_A) \leftarrow ZR_n,\ r_1 \leftarrow R_n,\ (y_0, s) = A(z, r_A),\ y_1 = A(s, r_1) :$$
$$\exists x',\ r_0,\ y_0 = f_n(x', r_0))\ \textit{and}\ y_1 = f_n(x', r_1))] - \mu(n)$$

3. *There exists a polynomial $p_{A'}$, such that for any polynomial $q > p_{A'}$:*

$$Pr[(z,r_A)\leftarrow ZR_n:$$

$$\frac{1}{q(n)}\leq Pr[r_1\leftarrow R_n,\ (y_0,s)=A'(z,r_A),\ y_1=A'(s,r_1,a_{A'}):\ \exists x',\ r_0,\ y_0=f_n(x',r_0)\ \textit{and}\ y_1=f_n(x',r_1)]$$

$$\leq\frac{1}{p_{A'}(n)}]\leq\mu(n)$$

then there is a deterministic polynomial-time machine, $\mathcal{K}$ such that for $n \in \mathbb{N}'$:

$$Pr[(z, r_A) \leftarrow ZR_n,\ r_1 \leftarrow R_n,\ (y_0, s) = A(z, r_A),\ y_1 = A(s, r_1),\ x = \mathcal{K}(z, r_A) :$$

$$\exists r_0, f_n(x, r_0) = y_0\ \textit{or}\ (\forall x'(\forall r_0,\ y_0 \neq f_n(x', r_0))\ \textit{or}\ y_1 \neq f_n(x', r_1))] > 1 - \mu(n). \quad (5.30)$$

Moreover, if $\mathbb{F}$ is efficiently computable and verifiable (as in Definition 2.5.1), then the converse is also true.

The proof is similar to that of Theorem 5.2.4. We use Lemma 5.2.2 to get a family $\mathbb{U}$ of machines. We claim that there is a member of this family that achieves negligible extraction error. If this were not to be the case, then for every member U_i there is a polynomial p_i such that U_i fails with probability at least $\frac{1}{p_i}$. Note that p_i may increase as i increases. However, by the third condition on A', consistency of A' is bounded from below by the inverse of a fixed polynomial independent of p_i. This is important because when we restrict the input distribution to where A' is consistent and $\mathbb{U}$ fails, A' remains consistent with noticeable probability. Consequently, we can apply Theorem 5.2.2 to get an extractor with noticeable success contradicting the lemma.

Proof. ($\Longrightarrow$)

The proof is almost the same as that of Theorem 5.2.4. The main difference is that each U_i is now assumed to fail with probability $\frac{1}{p_i(n)}$, where p_i depends on U_i (as opposed to some fixed polynomial). If we restrict the distribution to those elements on which the family, $\mathbb{U}$, fails collectively, then Lemma 5.2.2 states that all machines should fail on this distribution. However, *given no other conditions*, there is no guarantee that A will succeed noticeably on this restriction. On the other hand, reliable consistency gives us the guarantee that we need.

Formally, suppose for the purpose of contradiction that Eq. 5.30 does not hold. Then, there is a weakly-verifiable probabilistic function, $\mathbb{F}$, a PPT A, an infinite set of security parameters, $\mathbb{N}'$, a distribution on the auxiliary information and A's private input, $\mathbb{ZR}^1$, such that for any deterministic polynomial-time machine, $\mathcal{K}$, there is a polynomial, $p_{\mathcal{K}}$ and an infinite subset of security parameters $\mathbb{N}_{\mathcal{K}} \in \mathbb{N}'$ such that:

$$Pr[(z, r_A) \leftarrow ZR_n^1,\ r_1 \leftarrow R_n,\ (y_0, s) = A(z, r_A),\ y_1 = A(s, r_1), x = \mathcal{K}(z, r_A) :$$

$$(\forall r, f_n(x, r) \neq y_0 \text{ and } (\exists x',\ r_0,\ y_0 = f_n(x', r_0) \text{ and } y_1 = f_n(x', r_1))] \geq \frac{1}{p_{\mathcal{K}}(n)}. \quad (5.31)$$

Let A' be the PPT satisfying the three conditions described in this theorem and $\mathbb{ZR}^{A'} = \{ZR_n^{A'}\}_{n \in \mathbb{N}''}$ be the restriction of $\mathbb{ZR}^1$ to those elements on which A' succeeds with probability $\frac{1}{p_{A'}}$, Formally,

$$\mathbb{N}'' = \{n \in \mathbb{N}' : Pr[(z, r_A) \leftarrow ZR_n^1 : Pr[r_1 \leftarrow R_n,\ (y_0, s) = A'(z, r_A),\ y_1 = A(s, r_1) :$$

$$(\exists x,\ r_0,\ y_0 = f_n(x, r_0) \text{ and } y_1 = f_n(x, r_1))] \geq \frac{1}{p_{A'}(n)}] > 0\}.$$

Also, for any $n \in \mathbb{N}''$ and any (a, b):

$$Pr[(z, r_A) \leftarrow ZR_n^{A'} : (z, r_A) = (a, b)] =$$

$$\frac{1}{T_{ZR_n^{A'}}} Pr[(z, r_A) \leftarrow ZR_n^1 : (z, r_A) = (a, b) \text{ and}$$

$$Pr[r_1 \leftarrow R_n,\ (y_0, s) = A'(z, r_A),\ y_1 = A'(s, r_1, a_{A'}) :$$

$$\exists x, r_0,\ y_0 = f_n(x, r_0) \text{ and } y_1 = f_n(x, r_1)] \geq \frac{1}{p_{A'}(n)}],$$

where

$$T_{ZR_n^{A'}} = Pr[(z, r_A) \leftarrow ZR_n^1 : Pr[r_1 \leftarrow R_n,\ (y_0, s) = A'(z, r_A),\ y_1 = A(s, r_1) :$$

$$(\exists x,\ r_0,\ y_0 = f_n(x, r_0) \text{ and } y_1 = f_n(x, r_1))] \geq \frac{1}{p_{A'}(n)}].$$

Using conditions 1 and 2 of the theorem, Eq. 5.31 still holds if A is replaced with A'. Moreover, the same equation still holds if we replace $\mathbb{ZR}^1$ with $\mathbb{ZR}^{A'}$ because by condition

3, A' fails almost always on every element outside the support of $\mathbb{ZR}^{A'}$. Formally, for any deterministic polynomial-time machine, $\mathcal{K}$, there is a polynomial, $p_{\mathcal{K}}$ and an infinite subset of security parameters $\mathbb{N}_{\mathcal{K}} \in \mathbb{N}''$ such that:

$$Pr[(z, r_A) \leftarrow ZR_n^{A'},\ r_1 \leftarrow R_n,\ (y_0, s) = A'(z, r_A),\ y_1 = A'(s, r_1), x = \mathcal{K}(z, r_A) :$$

$$(\forall r, f_n(x, r) \neq y_0 \text{ and } (\exists x',\ r_0,\ y_0 = f_n(x', r_0) \text{ and } y_1 = f_n(x', r_1))] \geq \frac{1}{p_{\mathcal{K}}(n)}. \quad (5.32)$$

Let $\mathbb{G}$ and $\mathbb{V}_{\mathbb{ZR}^{A'}}$ be as defined in the proof of Theorem 5.2.3 on distribution $\mathbb{ZR}^{A'}$. Apply Lemma 5.2.2 on the parameters $\mathbb{G}$, $\mathbb{ZR}^{A'}$, and $\mathbb{V}_{\mathbb{ZR}^{A'}}$ to get the family $\mathbb{U} = \{U_i\}_{i \in \mathbb{N}}$ as described in that lemma.

By Eq. 5.32, for any U_i, there is a polynomial, p_i, and an infinite subset of security parameters $\mathbb{N}_{U_i^i} \subseteq \mathbb{N}''$ such that:

$$Pr[(z, r_A) \leftarrow ZR_n^{A'},\ (y_0, s) = A'(z, r_A),\ x = U_i^i(z, r_A) :$$

$$(\forall r, f_n(x, r) \neq y_0 \text{ and } (\exists x',\ r_0,\ y_0 = f_n(x', r_0) \text{ and } y_1 = f_n(x', r_1))] \geq \frac{1}{p_i(n)} \quad (5.33)$$

Let $\mathbb{ZR}^3 = \{ZR_n^3\}_{n \in \mathbb{N}_\infty}$ be the restriction of $\mathbb{ZR}^{A'}$ to those elements on which the family $\mathbb{U}$ fails (see proof of Theorem 5.2.3 for formal definitions of $\mathbb{ZR}^3$ and $\mathbb{N}_\infty$).

By construction, A' is consistent with probability $\frac{1}{p_{A'}}$ for any (z, r_A) in the support of $\mathbb{ZR}^{A'}$ (and consequently $\mathbb{ZR}^3$). Formally, for any $n \in \mathbb{N}_\infty$:

$$Pr[(z, r_A) \leftarrow ZR_n^3,\ r_1 \leftarrow R_n,\ (y_0, s) = A'(z, r_A),\ y_1 = A'(s, r_1), :$$

$$\exists x,\ r_0,\ y_0 = f_n(x, r_0) \text{ and } y_1 = f_n(x, r_1)] \geq \frac{1}{p_{A'}}. \quad (5.34)$$

Moreover, by construction we have for any i and sufficiently large $n \in \mathbb{N}_\infty$:

$$Pr[(z, r_A) \leftarrow ZR_n^3,\ (y_0, s) = A'(z, r_A),\ x = U_i^i(z, r_A) : \exists r, f_n(x, r) = y_0] = 0. \quad (5.35)$$

Eq. 5.34 with statement 2 of Theorem 5.2.2 imply the existence of a deterministic polynomial-time machine that extracts with nonnegligible probability over the distribution ZR_n^3. On the other hand, Eq. 5.35 with Lemma 5.2.2 (in particular, condition 2 of

this lemma with $\mathbb{ZR}^3$ as the distribution and $V_{A',\mathbb{ZR}^3}$ as a weak verifier) imply that every polynomial-time machine fails to extract on the same distribution, except with negligible probability. Consequently, a contradiction is reached and the forward direction of this theorem holds.

($\Longleftarrow$)

If $\mathbb{F}$ is efficiently computable and verifiable then the converse of the theorem is true. That is, if there is an extractor with negligible error then there is a PPT, A' satisfying the three conditions in this theorem. Specifically, let A' be the following machine. It computes $(y_0, s) = A(z, r_A)$ and outputs y_0, s. When it receives the challenge, r_1, it computes $x = \mathcal{K}(z, r_A)$ and checks whether $V(x, y_0) = 1$ (V is the deterministic verifier that always works, as in Definition 2.5.1). If so, it outputs $f_n(x, r_1)$. Otherwise, it returns $\perp$. A' satisfies condition 1 in a straightforward way. Moreover, A' satisfies condition 2 because A' is inconsistent only when $\mathcal{K}$ fails in inverting y_0. The latter event happens at most negligibly often when A is consistent. Finally, A' is consistent exactly when $\mathcal{K}$ recovers a correct preimage of y_0. Consequently, the consistency of A' is independent of the random coins r_1. Thus, if $p_{A'} = 2$, then the outer probability in condition 3 is 0. Formally, for any polynomial $q > 2$:

$$Pr[(z, r_A) \leftarrow ZR_n :$$

$$\frac{1}{q(n)} \leq$$

$$Pr[r_1 \leftarrow R_n,\ (y_0, s) = A'(z, r_A),\ y_1 = A'(s, r_1, a_{A'}) :\ \exists x',\ r_0,\ y_0 = f_n(x', r_0) \text{ and } y_1 = f_n(x', r_1)]$$

$$\leq \frac{1}{2}] = 0.$$

□

5.3 Noninteractive Extraction versus Obfuscation

A natural question that arises from the work of Section 5.2 on characterizing knowledge extraction in the interactive setting is, how does this translate to the noninteractive setting? We present similar results in this setting. However, the results are less informative and the implications seem weaker in the sense that functions seem to be more likely to satisfy statement 1 than statement 2 (which is our main objective). For instance, it seems to us that if we try to prove an alternative to Theorem 5.2.3 (amplification up to arbitrary small, yet negligible error) in the nonuniform and noninteractive setting, we need to build on an alternative version of Theorem 5.2.1, where all functions satisfy statement 1. In other words, functions do not satisfy the notion of extraction that seem

to us necessary for amplification in the nonuniform setting (we discuss this in more details later on in this section). Yet, for completeness, we present two theorems parallel to those in the interactive setting. The first one pertains to extraction with nonnegligible error and applies to the nonuniform and uniform setting. The second one deals with negligible extraction error but is in the uniform model only.

5.3.1 Weak Extraction

As in the interactive model, we observe that any function satisfies a certain "obfuscation" or "extraction" property. The obfuscation property says that there is a box, g that receives as input the description, k, of a function, f, and computes $f_k(x,r)$ for some x and r. On the other hand, nobody can recover x from this box. The extraction property means that any adversary that tries to output a point in the range of the function f_k, knows a corresponding preimage. As before, knowledge is captured by the existence of a nonblackbox extractor that computes a preimage with noticeable success. Formally,

Theorem 5.3.1. *Let $\mathbb{F} = \{\{F_k\}_{k\in K_n}\}_{n\in\mathbb{N}}$ be any family ensemble of probabilistic functions . Then, exactly one of the following two statements should hold:*

1. *There is an infinite subset of security parameters, $\mathbb{N}'$, a well-spread distribution, $\mathbb{K} = \{K_n\}_{n\in\mathbb{N}'}$, on the function key domain, and a probabilistic function, G such that for any nonuniform (respectively, uniform) polynomial-time machine, A:*

$$Pr[g \leftarrow G(1^N),\ k \leftarrow K_n,\ x = A(k,g)) : \exists r,\ g(k) = f_k(x,r)] \leq \mu(n) \tag{5.36}$$

and

$$Pr[g \leftarrow G(1^N),\ k \leftarrow K_n : \exists x,\ r,\ g(k) = f_k(x,r)], \tag{5.37}$$

is nonnegligible in n, and $g(.)$ runs in polynomial time.

2. *For any PPT A, any infinite subset of security parameters, $\mathbb{N}'$, any distribution, $\mathbb{ZR} = \{ZR_n\}_{n\in\mathbb{N}'}$, on auxiliary information and the private input of A, and any well-spread distribution, $\mathbb{K} = \{K_n\}_{n\in\mathbb{N}'}$, on the function key domain if:*

$$Pr[(z,r_A) \leftarrow ZR_n,\ k \leftarrow K_n,\ y = A(k,z,r_A) : \exists x,\ r,\ y = f_k(x,r)], \tag{5.38}$$

is nonnegligible in n, then there exists a nonuniform (respectively, uniform) polynomial-time machine, $\mathbb{K}$, such that:

$$Pr[(z, r_A) \leftarrow ZR_n,\ k \leftarrow K_n,\ y = A(k, z, r_A),\ x = \mathcal{K}(k, z, r_A) : \exists r,\ y = f_k(x, r)], \tag{5.39}$$

is nonnegligible in n.

Proof. Observe that statement 1 and 2 are mutually exclusive if we let $g(.)$ to be $A(., z, r_A)$ (i.e., a function of k) to prove one direction and $A(., z, r_A)$ to be $g(.)$ to prove the reverse direction. ($\Longrightarrow$)

Formally, suppose that statement 2 does not hold. Then, there exists a PPT, A, an infinite set of security parameters, $\mathbb{N}'$, a distribution over auxiliary information and A's private input, $\mathbb{ZR}$, a well-spread distribution, $\mathbb{K} = \{K_n\}_{n \in \mathbb{N}'}$, a polynomial, p_A, and an infinite subset of security parameters, $\mathbb{N}'' \subseteq \mathbb{N}'$ such that for all $n \in \mathbb{N}''$:

$$Pr[(z, r_A) \leftarrow ZR_n,\ k \leftarrow K_n,\ y = A(k, z, r_A) : \exists x,\ r,\ y = f_k(x, r)] \geq \frac{1}{p_A(n)}. \tag{5.40}$$

and

$$Pr[(z, r_A) \leftarrow ZR_n,\ k \leftarrow K_n,\ y = A(k, z, r_A),\ x = \mathcal{K}(k, z, r_A) : \exists r,\ y = f_k(x, r)] < \mu(n). \tag{5.41}$$

G samples z, r_A from ZR_n and outputs $g = A(., z, r_A)$. Using G, Eq. 5.36 and 5.37 follow from Eq. 5.41 and 5.40, respectively. ($\Longleftarrow$)

In the reverse direction, suppose that statement 1 holds. Let G be as defined in statement 1. Let A be the adversary with auxiliary information and randomness distribution identical to G (i.e., $g = z, r_A$) and $A(k, g) = g(k)$. Then, Eq. 5.38 and 5.39 follow directly from Eq. 5.37 and 5.36, respectively.

□

5.3.2 Amplifying Extraction

It seems that Theorem 5.3.1 is not sufficient to amplify extraction in the nonuniform setting. The problem seems to be the following. Suppose there is a family of nonuniform

extractors such that no other extractor can extract "considerably" better than this family. To prove that a member of this family succeeds except with arbitrary small probability, we restrict the distributions of ZR_n and K_n to elements on which this family fails. Statement 2 of Theorem 5.3.1 asserts the existence of an extractor that succeeds noticeably where this family fails. However, it may be the case that a member of this family fails on a polynomial fraction of the domain of ZR_n and of K_n. but for any polynomial fraction of ZR_n, there is a corresponding polynomial fraction of K_n on which it succeeds. Consequently, we can not restrict ZR_n and K_n *independently*. In other words, sampling an element from ZR_n is coupled with sampling an element from K_n. This coupling is not allowed by the current version of Theorem 5.3.1. On the other hand, if we allow this coupling, say, as follows: $Pr[(z, r_A, k) \leftarrow ZRK_n,\ y = A(k, z, r_A) : \exists x,\ r,\ y = f_k(x, r)]$. Then, every function that satisfies statement 2 is not one-way: If k is chosen with the code, g, (or with z, r_A), then $g_k(.)$ can simply contain $f_k(U_n, R_N)$ in the clear (and output it on input, k). Thus, if f is one-way, it is hard to recover a preimage.

In light of the previous discussion, we present knowledge amplification in the uniform setting only. As in the interactive setting, we assume that the function satisfies some form of weak verification. This notion is implied by public verification (as in Definition 2.5.1). Moreover, if a function is injective and has an extractor with arbitrary small error, then it satisfies weak verification.

One of the advantages of this result is that weak verification is sufficient not only for achieving arbitrary small yet noticeable extraction error (as in the interactive model), but also for extraction with negligible error.

Definition 5.3.1. *A family ensemble, $\mathbb{F} = \{\{F_k\}_{k\in K_n}\}_{n\in\mathbb{N}}$, satisfies weak verification if for every PPT, A (with input z, r_A), any distribution, $\mathbb{ZR} = \{ZR_n\}_{n\in\mathbb{N}'}$, on auxiliary information and the private input of A, any well-spread distribution, K_n, on the function description, and any polynomial p, there exists a uniform polynomial-time machine, $V_{A,ZR,K,p}$, such that for sufficiently large $n \in \mathbb{N}'$:*

$$Pr[(z, r_A) \leftarrow ZR_n,\ k \leftarrow K_n,\ y = A(k, z, r_A),\ :$$

$$(\exists x, r,\ V_{A,ZR,K,p}(x, k, z, r_A) \neq 1 \text{ and } f_k(x, r) = y$$

$$\text{or } \exists x, V_{A,ZR,K,p}(x, k, z, r_A) = 1 \text{ and } \forall r, f_n(x, r) \neq y)$$

$$and\ (\exists x, r,\ f_n(x,r) = y] < \frac{1}{p(n)}.$$

Theorem 5.3.2. *Let $\mathbb{F} = \{\{F_k\}_{k\in K_n}\}_{n\in\mathbb{N}}$ be any probabilistic family ensemble that satisfies statement 2 of Theorem 5.3.1 in the uniform setting. If $\mathbb{F}$ is weakly verifiable (as in Definition 5.3.1), then for any PPT A, any infinite set of security parameters, $\mathbb{N}'$, any well-spread distribution, K_n, on the function description, any distribution, $\mathbb{ZR} = \{ZR_n\}_{n\in\mathbb{N}'}$, on auxiliary information and the private input of A, there is polynomial-time machines, $\mathcal{K}$, such that for $n \in \mathbb{N}'$:*

$$Pr[(z, r_A) \leftarrow ZR_n,\ k \leftarrow K_n,\ y = A(k, z, r_A),\ x = \mathcal{K}(k, z, r_A) :$$

$$\exists r, f_k(x, r) = y\ \ or\ \forall x', r', y \neq f_k(x', r')] > 1 - \mu(n). \tag{5.42}$$

Moreover, this implication is an equivalence for injective functions.

Proof. ($\Longrightarrow$) Let G be the function that outputs k, z, r_A on input x if $A(k, z, r_A)$ computes an image of x under f_k (see proof of Theorem 5.2.3 for a similar formulation). Apply Lemma 5.2.2 on this function to get a family of uniform polynomial-time machines, $\mathbb{U}$. Suppose, for the purpose of contradiction, that Eq. 5.42 is not true. Then, every member, U_i, of $\mathbb{U}$ fails and A succeeds (in computing an valid image) with probability $\frac{1}{p_i(n)}$, where the polynomial p_i may depend on U_i. This can be rephrased as: For any U_i there is at least one element, z, r_A such that U_i fails and A succeeds with probability $\frac{1}{p_i(n)}$, where the probability is taken over K_n alone. Since K_n is well-spread, this can be rephrased as: for any U_i there is at least one element, z, r_A such that U_i fails and A succeeds over a super-polynomial number of keys taken from K_n. Now for any U_i, let $n = a_i$ be the smallest security parameter for which U_i fails with probability $\frac{1}{p_i(n)}$. Then, for any i and $n = a_i$, let z, r_A be the element on which U_i fails and A succeeds with probability $\frac{1}{p_i(n)}$ (where the probability is taken over K_n). Let ZR'_n be the distribution that samples this particular z, r_A only. Moreover, let K'_n be the uniform distribution on the elements of K_n such that $U_i(k, z, r_A)$ fails while $A(k, z, r_A)$ succeeds. As argued above, the support of K'_n is super-polynomial and consequently K'_n is well-spread. Now statement 2 of Theorem 5.3.1 with ZR'_n and K'_n implies the existence of an extractor that succeeds with nonnegligible probability where all members of $\mathbb{U}$ fail. This contradicts Lemma 5.2.2 (in particular, condition 2).

($\Longleftarrow$)

The converse of the above result is true for injective functions. That is, any extractor that fails with probability (even just) $\frac{1}{p}$ can be turned into a weak verifier that fails with the same probability. Specifically, Let $V_{A,\mathbb{ZR},K,p}(x,k,z,r_A) = 1$ if and only if $x = \mathcal{K}(k,z,r_A)$. Note that if $\mathcal{K}$ succeeds in computing a preimage, then $V_{A,\mathbb{ZR},p}$ behaves correctly on k, z, r_A and for any x (due to injection). □

Corollary 5.3.1. *An injective family ensemble, $\mathbb{F} = \{\{F_k\}_{k\in K_n}\}_{n\in\mathbb{N}}$, is extractable with negligible error if and only if it is extractable with arbitrary small, yet noticeable error.*

Proof. If $\mathbb{F}$ is injective and extractable with arbitrary small error then, as outlined in the proof of the previous theorem, $\mathbb{F}$ is also weakly verifiable. By Theorem 5.3.2, $\mathbb{F}$ is extractable with negligible error. □

Chapter 6

3-round Zero Knowledge

Summary: We show how a variant of extractable POW functions can be used to construct 3-round ZK arguments of knowledge and membership, using weaker knowledge assumptions than previously known results due to Hada and Tanaka (Crypto 1998) and Lepinski (M.S. Thesis, 2004). This also opens the door for constructing 3-round ZK arguments based on other assumptions.

6.1 Introduction

Zero-knowledge [GMR85] is one of the most fundamental notions of cryptography. This notion captures the idea of proving correctness of a statement *without revealing anything beyond its validity*. Zero-knowledge is usually manifested via a protocol between a prover and a verifier. In a zero-knowledge protocol, the task of a prover is to convince a verifier that a statement is correct but does not want to reveal anything beyond this.

Two properties of zero-knowledge protocols, namely soundness and zero-knowledge, protect against adversarial strategies from both the prover and verifier. Soundness guarantees that a dishonest prover can not convince the verifier of the correctness of an invalid statement. On the other hand, zero-knowledge insures that a malicious verifier does not learn anything beyond the validity of the statement. The latter property is formalized via the simulation paradigm. In other words, for any verifier, there is a simulator that can replicate the conversation with the prover without knowledge of a witness for the correctness of a statement.

This chapter is based on the paper [CD08a], which is a joint work with Ran Canetti. Note that [CD08a] contains some additional results that do not appear in this chapter.

Soundness can be required against efficient or unbounded provers. The first notion is called a ZK *argument* while the latter is called a ZK *proof*. Zero-knowledge can also be defined against efficient or unbounded verifiers. The first notion is called computational zero-knowledge while the latter is called statistical zero-knowledge. In this chapter, we use the term "zero-knowledge" to refer to computational zero-knowledge arguments with negligible error.

One of the major efficiency criteria of zero-knowledge protocols is round complexity, i.e., number of messages exchanged between the two parties. Lower bounds on round complexity include impossibility of 2-round zero-knowledge and 3-round *blackbox-simulation* zero-knowledge except for languages in BPP [GO94, GK96]. Current 3-round ZK arguments and proofs (with nonblackbox simulation) use strong and very specific number theoretic assumptions [HT98, HT99, Lep02, BP04b]. Therefore, constructing 3-round zero-knowledge based on weaker or general computational assumptions was posed as an open problem [Bar01].

6.1.1 Our Work

We apply a variant of extractable functions towards constructing 3-round ZK arguments and proofs of knowledge for any language in NP. This allows for abstracting from specific number theoretic assumptions and opens the door for basing 3-round ZK arguments on other computational assumptions.

At a high level, this construction uses the FLS technique [FLS99]. The FLS technique allows a machine with some extra information to convince the verifier of the validity of a statement *without knowledge of a witness*. Naturally, to preserve soundness this information is infeasible to compute by a prover interacting with a verifier. However, it is easy to compute by a simulator that has access to the private input of the verifier. Moreover, the verifier can not tell whether the interaction is fake or not.

In more detail, the construction is built on two main primitives: extractable POW functions and noninteractive witness-indistinguishable (WI) proofs [BOV03, GOS06]. We use a variant of extractable POW functions to give the simulator access to some extra information which is not available for the prover. Moreover, noninteractive WI proofs guarantee indistinguishability between a real communication and a simulated one.

Informally, the protocol starts with the prover sending an extractable POW function.

The verifier responds with a corresponding image of a random string. The protocol ends with the prover sending a noninteractive witness-indistinguishable (WI) proof that either the theorem is true or the prover knows a preimage of the verifier's message. Intuitively, this protocol is sound because the verifier's message completely hides its preimage. Thus, the (polynomially-bounded) prover can not efficiently recover a preimage. Consequently, if the verifier accepts the conversation then by the soundness of the WI proof, the theorem is true. On the other hand, this protocol is zero-knowledge because the verifier knows a preimage of its message. So, the simulator uses the extractor to recover a preimage and produces a WI proof using this preimage as a witness. For more detail, refer to Theorems 6.2.1 and 6.2.2.

As a concrete example, we can use Construction 3.3.2 in the protocol above. We remark that the knowledge assumptions required by Construction 3.3.2 and consequently by this protocol are weaker than the corresponding assumptions used for constructing 3-round ZK arguments in [HT98, HT99, BP04b]. Specifically, we eliminate the need for the second KE assumption in [HT99] and later updated in [BP04b]. See Corollaries 6.2.1 and 6.2.2 for more detail.

6.1.2 Related Work

Zero-knowledge proofs were introduced in [GMR85] with the first construction for any language in NP appearing in [GMW86]. Round-efficient (constant-round) constructions first appeared in [FS89, BCY89]. Bellare, Jakobsson, and Yung [BJY97] constructed 4-round zero-knowledge arguments for any language in NP from one-way functions. The last result is the most round-efficient zero-knowledge argument or proof from general computational assumptions. All of the above constant-round zero-knowledge protocols provide blackbox simulation in *expected* polynomial-time. Barak [Bar01] gave the first constant-round (5 rounds) ZK argument with strict polynomial-time simulation. Simulation of our protocols is in strict polynomial-time as well.

Goldreich and Oren [GO94] showed that 2-round zero-knowledge is possible only for languages in BPP while Goldreich and Krawczyk [GK96] proved that there is no *blackbox* zero-knowledge protocol for nontrivial languages. Both extraction and simulation of our protocols are nonblackbox and thus do not contradict these negative results.

Nonblackbox 3-round ZK arguments first appeared in [HT98, HT99, BP04b] while

Lepinski [Lep02] gave a 3-round ZK proof. All of the previous 3-round ZK protocols require very specific and nonstandard knowledge assumptions such that the KE assumption (Assumption 3.3.1) or the POK assumption (Assumption 3.3.4). On the other hand, our protocols are based on general computational assumptions without resorting to specific algebraic constructs.

6.2 Constructions

As mentioned in the introduction, we use an FLS-style technique [FLS99] on extractable functions to construct 3-round ZK arguments of membership and knowledge. Recall, at a high level, the prover utilizes a noninteractive WI arguments [GOS06] to prove that either the theorem is true or it knows a preimage of the challenge of the verifier. Clearly, to preserve soundness, the prover does not know a preimage of the challenge. On the other hand, a simulator can use an extractor to recover this preimage and then prove knowledge of it. Moreover, witness indistinguishability and perfect one-wayness guarantee indistinguishability between the simulated environment and the real one.

We present the argument of membership in Section 6.2.1 and proof of knowledge in Section 6.2.2.

6.2.1 Arguments of Membership

Recall, the prover chooses a new extractable POW function and sends its description to the verifier. The verifier chooses a random element, hashes it under the extractable function, and sends the hash to the prover in addition to a new extractable POW function. Finally, the prover verifies that the hash is valid. If so, it sends a noninteractive WI proof that either the theorem under consideration is true or the prover knows a preimage of the hash. In more detail, the prover sends in the last round a second hash (under the new function) and a noninteractive WI proof that either the original theorem is true or preimages of both hashes share a common substring.

Formally, let L be any NP language with the corresponding relation $\mathcal{R}_L$. Let $\mathbb{H}$ be a verifiable family ensemble. Define a new NP language:

$$L_{k_1,k_2} = \{x' = (x, y, s) : \exists w, (x, w) \in \mathcal{R}_L \text{ or } \exists u_1, u_2, V_{H_{k_1}}(u_1, y) = V_{H_{k_2}}((u_1, u_2), s) = 1\}.$$

$P(x,w)$		$V(x,z)$
$(k_1, z_{k_1}) \leftarrow K_n$		$(k_2, z_{k_2}) \leftarrow K_n$
	$\xrightarrow{k_1}$	
		$r \leftarrow R_n$
		$u \leftarrow U_n$
		$y = H_{k_1}(u, r)$
	$\xleftarrow{k_2, y}$	
$u_1, u_2 \leftarrow U_n, U_n$		
$s \leftarrow H_{k_2}((u_1, u_2, .)$		
$\pi = P'((x, y, s), w)$		
if $V_{range(\mathbb{H})}(y, z_{k_1}) \neq 1$		
send a uniform string and halt		
otherwise:	$\xrightarrow{s, \pi}$	
		if $V_{range(\mathbb{H})}(s, z_{k_2}) \neq 1$
		output 0 and halt
		otherwise:
		output $V'((x, y, s), \pi)$

Figure 6.1: A 3-round ZK Argument of Membership

Let $\mathcal{P}' = (P', V')$ be a noninteractive witness indistinguishable argument system for membership in L_{k_1,k_2}. The argument of membership is in Figure 6.1.

Theorem 6.2.1. *If there is an extractable and injective POW family ensemble with auxiliary information and range verification (as in Definitions 3.2.3, 2.5.5, 3.3.2, and 4.3.6) and noninteractive WI arguments exist for any language in NP (as in Definition 2.11.1, and where soundness is against nonuniform PPT provers), then* 3*-round ZK arguments of membership exist for any language in NP (as in Definition 2.8.1).*

Proof. For any language L in NP (with relation R_L), any extractable POW function, $\mathbb{H}$ and any noninteractive WI for the corresponding class of languages L_{k_1,k_2}, we argue that the protocol in Figure 6.1 is a zero-knowledge argument system for L.

Completeness. If P and V behave according to the protocol, then both y and s are valid images. So, $V_{range(\mathbb{H})}(y, z_{k_1}) = V_{range(\mathbb{H})}(s, z_{k_2}) = 1$. Consequently, neither P nor V aborts and the output of V is that of V'. Now, for any $x \in L$ and any (y, s), $(x, y, s) \in L_{k_1,k_2}$. So, by completeness of $\mathcal{P}'$, V' outputs 1.

Soundness. Let $x \notin L$. Since $\mathbb{H}$ is a POW function, then for any k_1 sent in the first round, the prover can not recover u from the message of the verifier. On the other hand, the prover knows a preimage of s by the extraction property on $\mathbb{H}$. Therefore, the preimage of s can not contain u without violating the one-way property on $\mathbb{H}$. The injective property guarantees that there is exactly one preimage of s so that if there is

one preimage that contains u, then this is the image that the extractor recovers. Consequently, if $x \notin L$, then the instance, $(x, y, s) \notin L_{k_1,k_2}$ with overwhelming probability. Thus, soundness follows from soundness on the underlying noninteractive WI argument system. Specifically, if the protocol in Figure 6.1 is not sound against a malicious prover $\hat{P}$, let $\hat{P}'$ be another malicious prover that defeats soundness of the underlying WI proof. $\hat{P}'$ receives x as auxiliary input (hence, the non-uniformity requirement), it simulates the communication of $\hat{P}(x)$ with an honest verifier and outputs (x, y, s, π), where y is in the message of the verifier and s, π is the last message of $\hat{P}$.

Zero-knowledge. For every PPT, $\hat{V}$, we show there is a nonblackbox simulator, S. S sends the first message exactly like an honest prover. Then, V^* sends the second message, k_2, y. The simulator uses $V_{range(\mathbb{H})}$ (and z_{k_1}) to verify that y belongs to the range of H_{k_1}. If y is valid, it uses the *nonblackbox* extractor for $\mathbb{H}$ and $\hat{V}$ to compute u. Then, it computes an image of (u, u') for some uniform u'. Finally, it uses the honest prover of the noninteractive WI argument system to convince the verifier using (u, u') as a witness for $x, H_{k_1}(u), H_{k_2}(u, u')$. By perfect one-wayness on $\mathbb{H}$:

$$x, z, k_1, k_2, H_{k_1}(u), H_{k_2}(u, u')$$

is computationally indistinguishable from

$$x, z, k_1, k_2, H_{k_1}(u), H_{k_2}(u', u'').$$

Moreover, by witness indistinguishability,

$$S(x, z) = x, z, k_1, k_2, H_{k_1}(u), H_{k_2}(u, u'), P'((x, H_{k_1}(u), H_{k_2}(u, u')), (u, u'))$$

is computationally indistinguishable from

$$x, z, k_1, k_2, H_{k_1}(u), H_{k_2}(u, u'), P'((x, H_{k_1}(u), H_{k_2}(u, u')), w).$$

Moreover, the latter distribution is indistinguishable from

$$< P(x, w), V(x, z) >= x, z, k_1, k_2, H_{k_1}(u), H_{k_2}(u', u''), P'((x, H_{k_1}(u), H_{k_2}(u', u'')), w)$$

The last claim is true because of the first indistinguishability claim in this paragraph. Suppose, for the purpose of contradiction, the latter claim is not true. Then, there is a pair $(x, w) \in R_L$, and a PPT D that distinguishes the last two distributions. Let A be a PPT that uses D to defeat the first indistinguishability claim in this paragraph. A receives z, (x, w), and $H_{k_1}(u)$ as auxiliary information. It also receives s which can be either $H_{k_2}(u, u')$ or $H_{k_2}(u', u'')$. A then computes $\pi = P'(x, H_{k_1}(u), s), w)$ and simulates D' on $x, z, k_1, k_2, H_{k_1}(u), s, \pi$. □

Corollary 6.2.1. *If the KE assumption holds with independent auxiliary information (as in Assumption 3.3.1) and the (strong) DDH assumption holds with auxiliary information (as in Assumption 3.3.2), and noninteractive WI arguments exist for any language in NP (as in Definition 2.11.1, and where soundness is against nonuniform PPT provers), then* 3*-round ZK arguments of membership exist for any language in NP (as in Definition 2.8.1).*

Proof. Use Construction 3.3.2 to get an extractable and injective POW function with auxiliary information and range verification. □

On the hardness assumption of $\mathbb{H}$. We emphasize that even though one-wayness of $\mathbb{H}$ is sufficient for soundness, it is not so for proving zero-knowledge. In more detail, when a (potentially malicious) prover, $\hat{P}$, interacts with the honest verifier, it is sufficient that $\mathbb{H}$ is only *one-way* against a uniform input. This insures that y does not reveal u. Thus, if $x \notin L$, then it is difficult for $\hat{P}$ to find a tuple, $(x, y, s) \in L_{k_1,k_2}$. On the other hand, one-wayness is not sufficient to prove zero-knowledge because s may reveal few bits about u_1, which is sufficient for a malicious verifier, $\hat{V}$, to detect whether s has a preimage that contains u. This detection allows $\hat{V}$ to distinguish interacting with an honest prover from interacting with a simulator.

On replacing injection with collision resistance. Injection is used in the proof of soundness of the protocol in Figure 6.1. It is used to guarantee equality of the preimage of y and the first substring of the preimage of s, *as recovered by the extractor*. Intuitively, collision resistance is sufficient for proving this claim. However, this seems not to be the case. One attempt at a proof using collision resistance is as follows. If the extractor recovers a preimage, (u', w), of s and there is another preimage u, w' (guaranteed by the assumption that $x \notin L$ but x, y, s is in L_{k_1,k_2}), we have an *almost* collision on s: simulate

$P(x,w)$		$V(x,z)$
$(k_1, z_{k_1}) \leftarrow K_n$		$(k_2, z_{k_2}) \leftarrow K_n$
	$\xrightarrow{k_1}$	
		$r \leftarrow R_n$
		$u \leftarrow U_n$
		$y = H_{k_1}(u, r)$
	$\xleftarrow{k_2, y}$	
$u_1, u_2, u_3 \leftarrow U_n, U_n, U_n$		
$r' \leftarrow R_n$		
$s \leftarrow H_{k_2}((u_1, u_2, .)$		
$v = H_{k_2}((w, u_3), r')$		
$\pi = P'((x, v, y, s), (w, u_3, r'))$		
if $V_{range(\mathbb{H})}(y, z_{k_1}) \neq 1$		
send a uniform string and halt		
otherwise:	$\xrightarrow{s, v, \pi}$	
		if $V_{range(\mathbb{H})}(s, z_{k_2}) \neq 1$ or
		$V_{range(\mathbb{H})}(v, z_{k_2}) \neq 1$:
		output 0 and halt
		otherwise:
		output $V'((x, v, y, s), \pi)$

Figure 6.2: A 3-round ZK Proof of Knowledge

the whole experiment, recover u from the honest adversary and u', w from the extractor. However, w' remains unknown. A potential remedy is to modify the protocol so that the prover sends an image, s', of w' as well. However, this still does not solve the problem. It may be possible the extractor for s' recovers a completely different preimage while w' remains hard to compute.

6.2.2 Proofs of Knowledge

The proof of knowledge protocol is very similar to the protocol in Figure 6.1. The only difference is that the prover has to prove knowledge of a witness. So, the prover computes an image, under an extractable function, of the witness and a uniform string and sends it to the verifier. This protocol uses a noninteractive WI system for the following language:

$$L'_{k_1,k_2} = \{x' = (x, v, y, s) : \exists w, u, r, (x, w) \in \mathcal{R}_L \text{ and } H_{k_2}(w, u, r) = v \text{ or}$$

$$\exists z_1, z_2, V_{\mathbb{H}}(z_1, y) = V_{\mathbb{H}}((z_1, z_2), s) = 1\}.$$

Theorem 6.2.2. *If there is an extractable and injective POW family ensemble with*

auxiliary information and range verification (as in Definitions 3.2.3, 2.5.5, 3.3.2, 4.3.6) and noninteractive WI arguments exist for any language in NP (as in Definition 2.11.1, and where soundness is against nonuniform PPT provers), then 3*-round ZK proofs of knowledge exist for any language in NP (as in Definition 2.9.1).*

Proof. For any language L in NP (with relation R_L), any extractable POW function, $\mathbb{H}$ and any noninteractive WI for the corresponding class of languages L'_{k_1,k_2}, we argue that the protocol in Figure 6.2 is a zero-knowledge proof of knowledge for L.

The proof of completeness, soundness, and zero-knowledge are very similar to that of Theorem 6.2.1 and are omitted here.

Proof of Knowledge. Let $\hat{P}$ be any malicious prover. If the honest verifier, V, accepts a conversation with $\hat{P}$, then by soundness, we know that $x \in L$ and v has a valid and unique (by injection) preimage under $\mathbb{H}$. By extraction on $\mathbb{H}$, there is an efficient extractor $\mathcal{K}_{\hat{P}}$ that recover the preimage, (w', u_3), of v. On the other hand, by indistinguishability and extraction on $\mathbb{H}$, it is infeasible for $\hat{P}$ to compute s such that there is u' and $V_{\mathbb{H}}((u, u'), s) = 1$, where $V_{\mathbb{H}}(u, y) = 1$. Otherwise, it is possible to invert y by simulating the whole protocol on y and using the extractor on s to recover u, u' (injection guarantees that the preimage extracted is u, u'). Also, by soundness of P', $x, v, y, s \in L'_{k_1,k_2}$. Since y and s do not share a preimage (or more precisely, s does not contain the preimage of y), then there is w, u'_3 such that $(x, w) \in R_L$ and $V_{\mathbb{H}}((w, u'_3), v) = 1$. By injection $w' = w$. □

Corollary 6.2.2. *If the KE assumption holds with independent auxiliary information (as in Assumption 3.3.1) and the (strong) DDH assumption holds with auxiliary information (as in Assumption 3.3.2), and noninteractive WI arguments exist for any language in NP (as in Definition 2.11.1, and where soundness is against nonuniform PPT provers), then* 3*-round ZK proofs of knowledge exist for any language in NP (as in Definition 2.8.1).*

We emphasize that both simulation and extraction in the previous protocol are non-blackbox. Thus, our results do not contradict the impossibility results in [GK96, BL04].

Chapter 7

Random Oracle Instantiation

Summary: We apply extractable functions towards Random Oracle instantiation in encryption schemes. Specifically, we convert a class of semantically secure and CCA2-secure encryption schemes in the Random Oracle model to concrete ones by simply replacing the Random Oracle with an extractable POW function, without much change in the logic of the original proof.

We initiate our study with an instantiation of a specific encryption scheme before studying instantiation of a more general class of encryption schemes, that includes schemes with no previously known instantiation such as OAEP (Bellare and Rogaway, EuroCrypt 1994).

Extractable functions are instrumental for these results because such functions can be used to capture, in the standard model, the "knowledge of queries" property that is so useful in the Random Oracle model.

7.1 Introduction

The Random Oracle (RO) methodology [FS86, BR93] consists of two steps. The first step involves designing a protocol and proving security in an idealized model called the RO model. In the RO model, all parties have oracle access to a public random function, O. The oracle answers are uniform and independent with only one constraint, specifically, that all answers to the same query are identical. The second step involves "moving" the protocol from this idealized model to the real world. This is done by "replacing"

This chapter is based on the paper [CD08a], which is a joint work with Ran Canetti. Note that [CD08a] contains some additional results that do not appear in this chapter.

the Random Oracle with a cryptographic hash function such as SHA1 [(FI93] or MD5 [Riv92]. In other words, every oracle call is replaced by a function call to some publicly known cryptographic hash function. This transformation is known as an instantiation of Random Oracles.

Although the first step of the RO methodology is rigorous, the second step remains a heuristic for the most part. While most results in this area provide proofs in the RO model, they lack even informal justification as to why the instantiated protocols may be secure. Such justification is of dire need given the fact that the RO methodology is not sound in general. Specifically, it was shown that there are schemes secure in the RO model without any secure instantiations [CGH98, MRH04, GK03]. Furthermore, there exist natural primitives that are realizable in the RO model but can not be realized at all in the standard model, regardless of the computational assumptions used [Nie02].

Given the general impossibility results mentioned above, one may resort to considering a proof in the RO model as a "stepping stone" towards a proof in the standard model. However, there is a severe flaw with this point of view: when it comes to security properties, proofs in the RO model use the Random Oracle somewhat like a Swiss Army knife. Random Oracles satisfy many cryptographic properties including collision resistance (it is hard to find two queries with the same RO answer, see Definition 2.5.2), uniformity (the answer to any query is uniformly distributed), unpredictability or correlation intractability [CGH98], programmability [Nie02] and knowledge of queries (any machine that computes $O(q)$ knows q). Furthermore, current work that use the RO methodology do not often highlight the specific properties of Random Oracles that are used or needed for the current proof. This makes translating a proof from the RO model to the standard model a harder task. And indeed, proofs in the RO model usually follow different lines from the corresponding ones in the standard model. This is contrary to the intuition behind the RO methodology, which is to use the randomness in the RO model to come up with simple proofs and then replace the Random Oracle by an appropriate function while maintaining the overall proof structure.

In light of the above discussion, it is interesting to identify specific properties of Random Oracles that are essential for the security of specific protocols. Once these properties are identified, it may then be possible to capture them with concrete functions that can be used to replace Random Oracles. Such an approach motivated the

introduction of perfectly one-way (POW) functions in [Can97] as functions that capture the hiding property of Random Oracles and that are then used to instantiate Random Oracles in a semantically-secure encryption scheme (see Definition 2.7.1). In another attempt, Boldyreva and Fischlin [BF06] introduce a strong variant of pseudorandom generators geared towards instantiating OAEP.

However, attempts at direct instantiation of encryption schemes secure against chosen ciphertext attacks (IND-CCA2, see Definition 2.7.2) have failed. It seems that one main problem is to translate a central property of Random Oracles, namely knowledge of queries, to the standard model. This property proves essential for the security proof in the RO model but it has not been previously formalized and captured by concrete functions.

7.1.1 Our Work

We use extractable functions to capture the "knowledge of queries" property mentioned above. Specifically, we use extractable POW functions not only to instantiate such schemes but also use a proof of security that follows similar logic as the original proof. The intended goal in this instantiation is not to try to achieve a more efficient construction than the existing ones in the literature but rather identify and realize the needed properties of the random oracle so that the proof of security remains the same in the standard model in both its logic and simplicity.

7.1.1.1 Using Extractable Functions to Instantiate a Specific Encryption Scheme

As mentioned before, POW functions are used in [Can97] to capture and realize semantic security of the encryption scheme in [BR93]. However, this is not sufficient for CCA2-security as POW functions may not guarantee extractability. So, an extractable POW function provides the missing link, namely preimage extraction, for replacing a Random Oracle by a POW function. Here, we use extractable POW functions to instantiate the second encryption scheme in [BR93] (recalled shortly), and translate the proof to the standard model in a straightforward way. This scheme uses a trapdoor permutation, M, and two Random Oracles, O_1, O_2, to encrypt a message, m, as $c = (M(r), O_1(r) \oplus m, O_2(r, m))$, where r is uniform. At a high level, it is CCA2-secure because the hiding

property of Random Oracles gives us semantic security while knowledge of queries gives us knowledge of plaintext (the latter property is what enables proving CCA2-security). Thus, if we replace the Random Oracle by an extractable POW function in the previous scheme we get a CCA2-secure encryption scheme in the standard model. This scheme can be either noninteractive or 3-round[1] depending on whether the POW function is noninteractively or interactively extractable.

7.1.1.2 Towards a General Instantiation of Encryption Schemes

We next address the question whether this methodology can be generalized to realize other encryption schemes in the RO model. However, we already know that the conventional instantiation is not secure in general [CGH98, MRH04]. Intuitively, the main reason why this is so is that efficient functions can not emulate unpredictability (answers to queries are uniform and independent of all other answers) and consistency (answers to same queries are the same for all parties) at the same time. So, we devise a different type of Random Oracle instantiation for a special class of encryption schemes, called **first-query hiding**. A first-query hiding encryption scheme is one where the first Random Oracle query made by the encryption algorithm is the same as that made by the decryption scheme and it is not revealed by the ciphertext. Even though first-query hiding encryption schemes are restricted, the negative results of [CGH98] still apply here.

In more detail, we device a construct that captures both unpredictability and consistency. The idea is simple. To achieve unpredictability, oracle answers are chosen uniformly and independently. To maintain consistency, these answers have to be secretly communicated to other parties. We emphasize that Random Oracles implicitly play this role and our construct tries to capture exactly this. Towards this end, we assume that the encryption and decryption algorithms, E and D, in the *original* scheme share a secret. Specifically, we require the original scheme to satisfy the first-query hiding property. Now, we can send the encryption of these answers using a symmetric-key encryption scheme with the first query as the secret key. This symmetric encryption scheme is built from POW functions. Using this construct, we instantiate any first-query hiding semantically-secure encryption scheme in the RO model. Moreover, extractable POW functions can be used to convert these schemes into CCA2-secure schemes. See

[1] Refer to Section 7.2.3.1 for definitions of 3-round encryption.

Section 7.3 for more detail.

A new Instantiation. We emphasize that we use a new type of instantiation to realize both unpredictability and consistency. Clearly, the conventional instantiation that replaces every Random Oracle call by a function call is simpler. However, our deviation from this tradition is necessary. This is so because we realize a class of schemes that includes schemes provably uninstantiable under the conventional method. Specifically, the transformation in [CGH98] can use any first-query hiding scheme to yield a scheme that is not instantiable in the standard way but our technique works for such a scheme as well (refer to Section 7.3.3).

7.1.1.3 Instantiating OAEP

OAEP [BR94] is a commonly used and standardized encryption scheme. In spite of its popularity, it is not previously known whether it has any CCA2-secure or even semantically-secure instantiation. Even though OAEP does not satisfy the first-query hiding requirement, our results imply that OAEP has both a semantically-secure and CCA2-secure instantiation (see Section 7.4). These results utilize the assumption that the trapdoor permutation used in OAEP is partially one-way [FOPS01], i.e., it does not reveal the first part of the input. We emphasize that these instantiations are of a type different from the conventional one. We note that Boldyreva and Fischlin [BF06] use the traditional instantiation but their (full instantiation) result is limited to CPA-security of unknown random plaintext.

On the number of decryption queries. In CCA2 encryption, an adversary is allowed to ask a polynomial number of decryption queries. The only restriction is that these queries do not include the challenge ciphertext. Whereas this requirement is met by all 3-round instantiations, our noninteractive instantiations allow only a constant number of decryption queries. The reason seems to be the dependency of the extractor on the adversary. This implies that each decryption query may potentially require a different extractor.

7.1.1.4 On the Connection to Other Approaches and CCA2 Schemes

We remark that generic transformations from any semantically-secure scheme to a CCA2-secure one have been studied before [DDN00, Sah99]. Also, the KE assumption (see

Assumption 3.3.1) has been used to prove that certain encryption schemes are plaintext-aware, which when coupled with semantic security gives CCA-secure schemes [BP04b, Den06]. Moreover, Katz [Kat03] used the notion of proofs of plaintext knowledge to construct efficient 3-round CCA2-secure schemes. We emphasize that the contributions of this work are not in giving better or more efficient constructions than existing ones in the literature, but rather in the methodology of replacing Random Oracles as described above.

7.1.2 Organization

We instantiate the encryption scheme of [BR93] in Section 7.2. Also, we instantiate first-query hiding encryption schemes in Section 7.3, and apply these results to OAEP in Section 7.4.

7.2 Instantiation of a Specific Encryption Scheme

We use extractable POW functions to instantiate Random Oracles in the second encryption scheme of [BR93] while maintaining a similar proof of security. Extractable POW functions allow us to do so because they capture two properties of Random Oracles essential for the original proof, namely, pseudorandomness and knowledge of queries.

We recall the original scheme and highlights of its proof in Section 7.2.1 and instantiate it in Sections 7.2.2 (noninteractive) and 7.2.3 (interactive).

7.2.1 The Original Scheme

The original construction uses a family ensemble of trapdoor permutations, $\mathbb{M}$, with key space PK_n and trapdoor space SK_n, and two random oracles O_1 and O_2. The encryption of a message, m, is $c = M_{pk}(q), O_1(q) \oplus m, O_2(m, q)$, where q is uniform. Formally, encryption and decryption are as follows.

Construction 7.2.1 (The Original Scheme, [BR93]). *Let $\mathbb{M} = \{\mathbf{M}^n\}_{n\in\mathbb{N}}$ be a family ensemble of trapdoor permutations with key space PK_n and trapdoor SK_n, where $M_{pk} : \{0,1\}^n \to \{0,1\}^n$. Let O_1 be a random function from $\{0,1\}^n$ to $\{0,1\}^{l(n)}$ and O_2 be another random function from $\{0,1\}^{n+l(n)}$ to $\{0,1\}^{l(n+l(n))}$ for some polynomial l:*

- $E^{O_1,O_2}(m, pk)$ *selects* q *uniformly from* $\{0,1\}^n$ *and returns* $M_{pk}(q), O_1(q) \oplus m, O_2(m, q)$.

- $D^{O_1,O_2}(c = (c_1, c_2, c_3), sk)$ *computes* $q = M_{sk}(c_1)$, $m = c_2 \oplus O_1(q)$, *and returns* m *if* $O_2(m, q) = c_3$. *Otherwise, it returns* $\perp$ *(*c *is invalid).*

Informally, this construction is IND-CCA2 because it is IND-CPA and the decryption oracle does not help the adversary. In more detail, the proof assumes the existence of trapdoor permutation and consists of a reduction from the security of the construction to the security of the trapdoor permutation. Specifically, an adversary, A, that defeats IND-CCA2 can be turned into an adversary, B, that inverts the trapdoor permutation. To invert y, B runs A and simulates both the Random Oracle and decryption oracle. If A queries the Random Oracle on q or q, m such that $M_{pk}(q) = y$, B has found a preimage. Otherwise, it chooses an answer uniformly and returns it to A. Whenever A makes a decryption query, c_1, c_2, c_3, B checks if A has already made two Random Oracle queries, q and q, m satisfying the conditions $M_{pk}(q) = c_1, O_1(q) \oplus m = c_2, O_2(m, q) = c_3$. If so, B returns m. Moreover, when A outputs a message pair m_0, m_1, B responds with the challenge ciphertext y, c_2, c_3 where c_2 and c_3 are chosen uniformly. Then, B continues to run A as described above.

It is shown [BR93] that B has a noticeable success in inverting y. This is so because if A does not query the Random Oracle on q and (m, q) and does not query the decryption oracle on y, c_2', c_3', its advantage is zero. Moreover, if A asks for the decryption of y, c_2', c_3' without asking O_2 for the image of $(q, O_1(q) \oplus c_2')$, its advantage remains negligible. This argument can be rephrased as: Without access to a decryption oracle, A has a negligible advantage because M is one-way. On the other hand, any *valid* decryption query, c_1, c_2, c_3, that A makes must be preceded by two Random Oracle queries, $M_{sk}(c_1)$ and $(M_{sk}(c_1), O_1(M_{sk}(c_1)) \oplus c_2)$. However, if A makes any of these two queries it can compute the plaintext on its own.

Jumping ahead, the proof of our instantiation follows similar lines. We first prove that A can not achieve noticeable advantage without access to a decryption oracle, i.e., the construction is semantically secure. Then, we prove that the decryption oracle can be removed without changing the advantage of A because it knows the plaintext of its decryption queries.

7.2.2 Noninteractive Instantiation

We replace both random oracles O_1 and O_2 in Construction 7.2.1 with an extractable *(with dependent auxiliary information)* and pseudorandom POW (with public randomness and auxiliary information) function. This instantiation maintains security and a similar proof if the adversary is restricted to ask a constant number of decryption queries. At a very high level, the proof uses perfect one-wayness to prove semantic security and extraction to reduce CCA2-security to semantic security. The formal construction and proof follow.

Construction 7.2.2. *Let $\mathbb{M}$ is a trapdoor permutation (with key space PK_n and trapdoor SK_n) and $\mathbb{H}$ be a verifiable family ensemble. Define the encryption scheme, (G, E, D) as follows:*

- $G(1^n) = (k_1, k_2, pk, sk)$, *where* $pk, sk \leftarrow PK_n, SK_n$ *and* $k_1, k_2 \leftarrow K_n$.
- $E(m, pk' = (pk, k_1, k_2)) = r_1, M_{pk}(q), y \oplus m, H_{k_2}((q, m, r_1), r_2)$, *where* r_1, r_2, q *are uniform and* $H_{k_1}(q, r_1) = r_1, y$.
- $D(c = (r_1, c_1, c_2, c_3 = (r_2, c_3')), sk' = (sk, k_1, k_2))$ *computes* $q = M_{sk}(c_1)$, $r_1, y = H_{k_1}(q, r_1)$, $m = c_2 \oplus y$, *and check if* $c_3 = H_{k_2}((q, m, r_1), r_2)$. *If so, it outputs* m, *otherwise* $\perp$ *(c is invalid).*

Theorem 7.2.1. *If there exists a family ensemble, $\mathbb{H}$, that satisfies preimage extraction (as in Definition 3.2.5), pseudorandomness with auxiliary input (as in Definition 2.5.6), collision resistance (as in Definition 2.5.2), and public randomness, and there exists a family of trapdoor permutations, then Construction 7.2.2 is IND-CCA2 (as in Definition 2.7.2) against a constant number of decryption queries.*

Proof. As mentioned before, we show that Construction 7.2.2 is IND-CPA based on the perfect one-wayness of $\mathbb{H}$ and one-wayness of $\mathbb{M}$. Then, we use preimage extraction to show that IND-CPA implies IND-CCA2 for this scheme.

Construction 7.2.2 is IND-CPA (as in Definition 2.7.1)

Let $A = (A_1, A_2)$ be any PPT that tries to defeat IND-CPA of Construction 7.2.2. Let m_0, m_1 be any message pair that A_1 produces. Since $M_{pk}(q)$ is one-way in q, then by pseudorandomness of $\mathbb{H}$ with auxiliary input, we have $r_1, M_pk(q), y$ (where $H_{k_1}(q, r_1) = r_1, y)$) is computationally indistinguishable from $r_1, M_{pk}(q), U_{|H_{k_1}(q,r_1)|-|r_1|}$.

Consequently, for any $m \in \{m_0, m_1\}$, $r_1, M_{pk}(q), y \oplus m$ is also computationally indistinguishable from $r_1, M_{pk}(q), U_{|H_{k_1}(q,r_1)|-|r|} \oplus m$. Otherwise, one can distinguish the former two distributions: run A to compute m_0, m_1, choose one of the two message at random, xor it to the third string in the input, and then run the distinguisher for the latter two distribution. Moreover, for any m, we have $r_1, M_{pk}(q), U_{|H_{k_1}(q,r_1)|-|r|} \oplus m \equiv r_1, M_{pk}(q), U_{|H_{k_1}(q,r_1)|-|r|}$. Thus, $r_1, M_pk(q), y \oplus m$ is indistinguishable from $r_1, M_{pk}(q)$, $U_{|H_{k_1}(q,r_1)|-|r|}$. This implies that $r_1, M_pk(q), y \oplus m$ is one-way in q, m. Using again the fact the $\mathbb{H}$ is POW with auxiliary information, we have $r_1, M_pk(q), y \oplus m, H_{k_2}((q, m, r_1),$ $r_2)$ is indistinguishable from $r_1, M_pk(q), y \oplus m, U_{|H_{k_2}((q,m,r_1),r_2)|}$. By the previous argument, the latter distribution is indistinguishable from $r_1, M_{pk}(q), U_{|H_{k_1}(q,r_1)|-|r|}$, $U_{|H_{k_2}((q,m,r_1),r_2)|}$. Consequently, for any $m \in \{m_0, m_1\}$ that A_1 outputs, $E(m, pk') = r_1, M_pk(q), y \oplus m, H_{k_2}((q, m, r_1), r_2)$ is indistinguishable from $r_1, M_{pk}(q), U_{|H_{k_1}(q,r_1)|-|r|}$, $U_{|H_{k_2}((q,m,r_1),r_2)|}$. The result follows.

Construction 7.2.2 is IND-CCA2 (as in Definition 2.7.2)

Now, we use the assumption that $\mathbb{H}$ is extractable and that this construction is IND-CPA to conclude that it is IND-CCA2 with a constant number of decryption queries. Informally, we show how to construct from any machine, $A = (A_1, A_2)$, that breaks IND-CCA2 another one, $B = (B_1, B_2)$, that breaks IND-CPA. B behaves very much like A except with things to do with decryption queries. Since in the IND-CPA setting, adversaries do not have access to a decryption oracle, B has to somehow answer A's decryption queries on its own. Ofcourse, the way to do that is by utilizing extractability to find the desired preimage. Once a preimage is found, decryption queries can be correctly answered. In other words, B will simulate A until a decryption query occurs. Then, the simulation is paused, the extractor runs to find a preimage, a decryption answer is computed, and the simulation resumes again.

Formally, let $A = (A_1, A_2)$ be any adversary that defeats CCA2-security of Construction 7.2.2. Let l be a constant bounding the number of decryption queries that A makes. Let $c = r_1, c_1, c_2, c_3$ be the ciphertext that A receives. We first show that for any valid decryption query, d, that A makes, can not contain c_3. This is because if d contains c_3 then $d = c$ (which is not permitted). In more detail, if d contains c_3, and $d \neq c$, then there exists a pair q, m, r_1 and q', m', r_1' such that $(q, m, r_1) \neq (q', m', r_1')$ and $V_{\mathbb{H}}(q, m, r_1, c_3) = V_{\mathbb{H}}(q', m', r_1', c_3) = 1$. However, this contradicts collision resistance (a

collision resistance adversary can simulate the whole experiment with knowledge of sk to recover q, m, r_1 and q', m', r_1').

Let $A^1, ..., A^l$ be a sequence of machines, where A^i simulates A until the ith query, then it outputs the last substring of the ith query and halts. By extraction, there is a corresponding extractor, $\mathbb{K}_{A^i}$ for A^i. In more detail, A^i is defined inductively: A^i simulates A^{i-1}, which stops at the $(i-1)$th query, then A^i runs $\mathbb{K}_{A^{i-1}}$ to compute the decryption and continues to simulate A until the ith query.

Now, B runs A^l (which asks only 1 decryption query), and uses $\mathbb{K}_{A^l}$ to answer this decryption query. Note that the answers returned by the extractor should be the same as those returned by the decryption oracle except with negligible probability. Otherwise, collision resistance is violated: simulate the whole experiment with knowledge of the decryption key. Then, use the decryption key to recover one preimage of c_3 and the extractor to recover a different one. Consequently, except with negligible probability, the view of A is the same when simulated by B as when interacting with a decryption oracle. Thus, B defeats semantic security with probability overwhelmingly close to the probability of A defeating CCA2-security. □

Towards strengthening Theorem 7.2.1. Observe from the proof of Theorem 7.2.1 that extraction is used only on the last substring, c_3, of the ciphertext while pseudorandomness is needed only for the second substring, c_2, to mask the plaintext, m. Thus, Theorem 7.2.1 can be strengthened by using two different POW functions. The first one is a pseudorandom POW function with auxiliary information (as in Definition 2.5.6) but not necessarily extractable. This function is used in place of H_{k_1} in Construction 7.2.2. The second function is an extractable POW function (as in Definitions 2.5.5 and 3.2.5) and replaces H_{k_2} in Construction 7.2.2.

7.2.3 Interactive Instantiation

We use interactively-extractable POW functions to instantiate Construction 7.2.1 and get an interactive encryption scheme. We formalize the notion of interactive encryption in Section 7.2.3.1 and give the construction in Section 7.2.3.2.

7.2.3.1 Interactive Encryption

In an interactive (3-round) encryption scheme, the encryption algorithm engages in a 3-round communication with the decryption algorithm in order to transmit the plaintext securely. In other words, if the communication is consistent, then after the interaction is over, the decryption algorithm is able to output the intended plaintext. In this model, the ciphertext consists of the interaction between the encryption and decryption algorithm, denoted by $< E(m, pk), D(sk) >$. Indistinguishability under a chosen plaintext attack means the adversary can not tell by observing the ciphertext (communication) which message the encryption algorithm is transmitting. Formally,

Definition 7.2.1 (Interactive IND-CPA). *An interactive public key encryption scheme,* (G, E, D)*, is called IND-CPA if for any PPT pair* (A_1, A_2)*:*

$$|Pr[(pk, sk) \leftarrow G(1^n),\ (m_0, m_1, s) \leftarrow A_1(pk),$$

$$c \leftarrow< E(m_0, pk), D(sk) >,\ b \leftarrow A_2(s, c) : b = 1]\ -$$

$$Pr[(pk, sk) \leftarrow G(1^n),\ (m_0, m_1, s) \leftarrow A_1(pk),$$

$$c \leftarrow< E(m_1, pk), D(sk) >,\ b \leftarrow A_2(s, c) : b = 1]| \leq \mu(n),$$

where $< E(m, pk), D(sk) >$ *is the distribution over possible messages communicated between* E *and* D*.*

We emphasize that unlike the common notion of encryption, where decryption is deterministic, this notion allows for probabilistic decryption. In particular, the probability above is taken over the random coins of D as well.

Our notion of interactive CCA2 security assumes the existence of "phonecall-type" channel between the honest encryption and decryption algorithm when the challenge ciphertext is computed. This definition differs from the one in [Kat03] in which the adversary may stage a man-in-the-middle attack. However, digital signatures can be introduced into our schemes to achieve the stronger definition, e.g., by using signatures as in the interactive encryption scheme of [DDN00]. Nevertheless, for clarity and to focus on the applications and usage of extractable POW functions, we avoid using signatures and settle for the weaker definition.

Definition 7.2.2 (Interactive IND-CCA2). *An interactive public key encryption scheme, (G, E, D), is called IND-CCA2 if for any PPT pair $(A_1^{D(sk)}, A_2^{D(sk)})$:*

$$|Pr[(pk, sk) \leftarrow G(1^n),\ (m_0, m_1, s) \leftarrow A_1^{D(sk)}(pk),$$

$$c \leftarrow< E(m_0, pk), D(sk) >,\ b \leftarrow A_2^{D(sk)}(s, c) : b = 1]\ -$$

$$Pr[(pk, sk) \leftarrow G(1^n),\ (m_0, m_1, s) \leftarrow A_1^{D(sk)}(pk),$$

$$c \leftarrow< E(m_1, pk), D(sk) >,\ b \leftarrow A_2^{D(sk)}(s, c) : b = 1]| < \mu(n),$$

where $< E(m, pk), D(sk) >$ is the distribution over possible messages communicated between E and D. Moreover, we assume that $c' \neq c$ for $c' \leftarrow< A_2(s, c), D(sk) >$.

In the previous definition A is prohibited from replaying the same communication with D as in c. Moreover, this may not be possible, even if permitted, against probabilistic decryption.

7.2.3.2 The Construction

The idea behind this instantiation is to make use of interaction to verify that the sender actually knows q. This utilizes the fact that $\mathbb{H}$ satisfies interactive extraction. So that any adversary communicating with the decryption oracle knows what the plaintext is. Hence, the decryption oracle does not really help the adversary. Therefore, IND-CCA2 can be reduced to IND-CPA. Since this construction is IND-CPA, it must be IND-CCA2.

To encrypt a message, m, E sends an image of a uniform string, q, in the first round. D responds by sending random strings $r_1, ..., r_n$. In the last round, E sends n images of q using $r_1, ..., r_n$ as random coins for $\mathbb{H}$. E also sends the ciphertext of m using the original construction (with $\mathbb{H}$ in place of the Random Oracle) with the same q as the one used in the first round. We note that the first two messages are independent of the plaintext and thus can be sent ahead of time.

The formal construction appears in Figure 7.1 and the claim is in Theorem 7.2.2. We emphasize that we don't restrict the number of decryption queries the adversary makes (as in Theorem 7.2.1). This is so because we use the universal blackbox extractor of Definition 4.2.6.

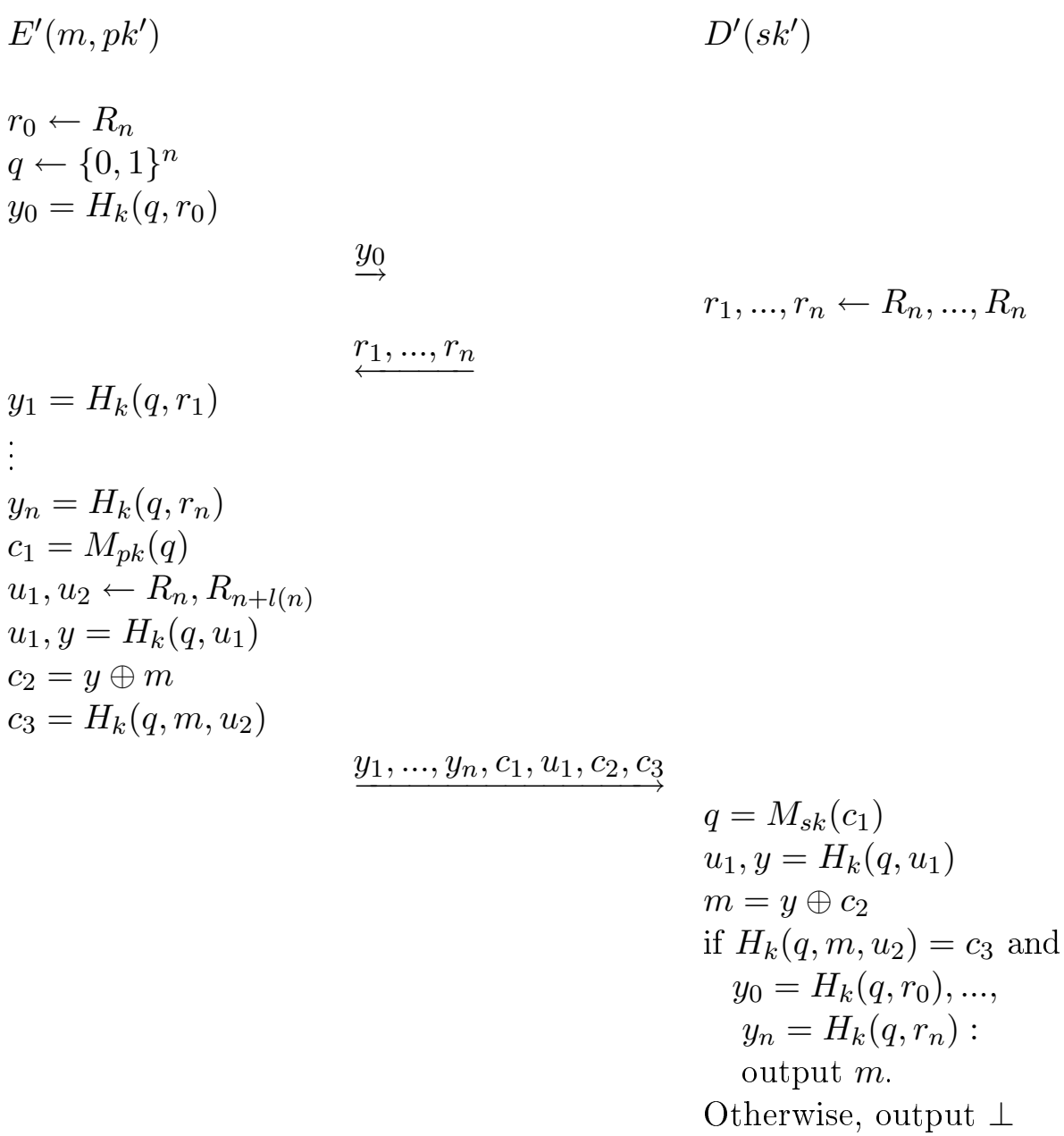

Figure 7.1: Interactive Instantiation of the Second Encryption Scheme in [BR93]

Theorem 7.2.2. *If there exists a family ensemble that satisfies preimage extraction (as in Definition 4.2.6), pseudorandomness with auxiliary input (as in Definition 2.5.6), collision resistance (as in Definition 2.5.2), and public randomness, and there exists a family of trapdoor permutations, then the construction in Figure 7.1 is IND-CCA2 (as in Definition 7.2.2).*

Proof. As mentioned before, this proof follows similar lines as the original proof. First, we show that without a decryption oracle, the adversary can not possibly have a noticeable advantage. In other words, we show the construction is IND-CPA. Second, we argue that a decryption oracle does not help because an adversary can compute on its own answers to its queries. Pseudorandomness with auxiliary information allows us to prove IND-CPA and extractability proves the second part.

Let $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$ be an extractable POW with auxiliary input and public randomness. Let $\mathbb{M} = \{\mathbf{M}^n\}_{n\in\mathbb{N}}$ be any family ensemble of trapdoor permutations.

The construction satisfies IND-CPA (as in Definition 7.2.1) This proof is very similar to the corresponding part in the proof of Theorem 7.2.1. We use $\mathbb{H}$'s pseudorandomness with auxiliary input to show that this Construction is IND-CPA. The proof consists of three steps. First, we consider q as our input to $\mathbb{H}$ and $M_{pk}(q)$ as auxiliary input about q. Since $\mathbb{H}$ is pseudorandom with respect to auxiliary information, $M_{pk}(q), H_k(q, u_1)$ (and thus, $M_{pk}(q), H_k(q, u_1) \oplus m_b$, where $b \in \{0, 1\}$) is indistinguishable from $M_{pk}(q), U_{l(n)}$. This implies that $M_{pk}(q), H_k(q, u_1) \oplus m_b$ is uninvertible in q. Second, we consider q, m_b as our input to $\mathbb{H}$. Again, by $\mathbb{H}$'s pseudorandomness with auxiliary information, $M_{pk}(q), H_k(q, u_1) \oplus m_b, H_k(q, m_b, u_2)$ is indistinguishable from $M_{pk}(q), H_k(q, u_1) \oplus m_b, U_{l(n+l(n))}$. Finally, by a similar argument, the latter distribution is indistinguishable from $M_{pk}(q), U_{l(n)}, U_{l(n+l(n))}$. We conclude that this construction is IND-CPA.

This proof can be interpreted as: If the construction is not IND-CPA then, by the properties on $\mathbb{H}$, $\mathbb{M}$ is not one-way and thus not a trapdoor permutation.

The construction satisfies IND-CCA2 (as in Definition 7.2.2)

Now, we use the assumption that $\mathbb{H}$ is extractable and that this construction is IND-CPA to conclude that it is IND-CCA2. Informally, we show how to construct from any machine, $A = (A_1, A_2)$, that breaks IND-CCA2 another one, $B = (B_1, B_2)$, that breaks IND-CPA. B behaves very much like A except with things to do with decryption queries.

Since in the IND-CPA setting, adversaries do not have access to a decryption oracle, B has to somehow answer A's decryption queries on its own. Ofcourse, the way to do that is by utilizing extractability to find the desired preimage. Once a preimage is found, decryption queries can be easily and correctly answered. In other words, B will simulate A until a decryption query occurs. Then, the simulation is paused, the extractor runs to find a preimage, a decryption answer is computed, and the simulation resumes again.

A slight complication arises due to the nature of our extractor. Since the extractor is known to succeed with probability at least $1 - \frac{1}{p} - \mu$, for some polynomial p, invoking it multiple times will lower its chances of answering all queries. In particular, if we invoke it $np(n)$ times, then it *may* have a negligible chance in answering all queries correctly. This means simulation of a decryption oracle may almost always differ from a real one. To avoid that, we allow our extractor's running time to depend on the number of queries that A makes.

Formally, suppose, for the purpose of contradiction, that this construction does not satisfy Definition 7.2.2. Then, there exists a PPT pair $A = (A_1, A_2)$, an infinite set of security parameters $\mathbb{N}' \subseteq \mathbb{N}$, and a polynomial p, such that:

$$Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s) \leftarrow A_1^{D'(sk')}(pk'),$$

$$c \leftarrow< E'(m_0, pk'), D'(sk') >,\ b \leftarrow A_2^{D'(sk')}(s, c) : b = 1] \ -$$

$$Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s) \leftarrow A_1^{D'(sk')}(pk'),$$

$$c \leftarrow< E'(m_1, pk'), D'(sk') >,\ b \leftarrow A_2^{D'(sk')}(s, c) : b = 1] > \frac{1}{p(n)}. \tag{7.1}$$

Let p_A be a polynomial bounding the running time of A. Then, A can ask at most p_A queries. Without loss of generality, we assume that $\forall n,\ p_A(n) \geq p(n)$. Let $\mathcal{K}$ be a PPT capable of extraction with probability at least $1 - \frac{1}{6p_A^2} - \mu$. Consider the following PPT pair $B = (B_1, B_2)$ that attacks IND-CPA of the same construction. $B_1(pk')$ simulates $A_1^{SIM(pk')}(pk')$ and outputs what A_1 does. Similarly, $B_2(s, c)$ simulates $A_2^{SIM(pk')}(s, c)$. Next, we describe SIM in more detail.

SIM is a PPT that tries to answer A's interactive decryption queries. It utilizes $\mathcal{K}$, as defined earlier. Informally, everytime SIM is invoked, $\mathcal{K}$ interacts (with rewinding) with A to compute a preimage, q. To find the corresponding plaintext, SIM checks

whether q, as computed by $\mathcal{K}$, is consistent with the communication that took place. If so, it computes the plaintext. Otherwise, it returns $\perp$. Formally, SIM is defined in Algorithm 7.2.1.

input : pk'
interaction: with an external PPT, A

$q \leftarrow \mathcal{K}^A(k)$;
receive y_0;
$r_1, ..., r_n \leftarrow R_n, ..., R_n$;
send $r_1, ..., r_n$;
receive $y_1, ..., y_n, c_1, u_1, c_2, c_3$;
$H_k(q, u_1) = u_1, y$;
$m = y \oplus c_2$;
if $\forall i \leq n,\ H_k(q, r_i) = y_i$ **and** $M_{pk}(q) = c_1$ **and** $H_k(q, m, u_2) = c_3$ **then**
    return m;
else
    return $\perp$;
end

Algorithm 7.2.1: SIM

Analysis

We show that using SIM as a decryption oracle instead of D does not change the output of A except with probability at most $\frac{1}{3p} + \mu$. Combining this claim with Eq. 7.1, we have that A^{SIM} (and consequently B) breaks IND-CPA with probability at least $\frac{1}{3p} - \mu$. Formally, we need to show for any $b \in \{0, 1\}$ and all n:

$$|Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s) \leftarrow A_1^{SIM(pk')}(pk'),\ c \leftarrow < E'(m_b, pk'), D'(sk') >,$$

$$b \leftarrow A_2^{SIM(pk')}(s, c) : b = 1] -$$

$$Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s) \leftarrow A_1^{D'(sk')}(pk'),\ c \leftarrow < E'(m_b, pk'), D'(sk') >,$$

$$b \leftarrow A_2^{D'(sk')}(s, c) : b = 1]| \leq \frac{1}{3p(n)} + \mu(n), \tag{7.2}$$

To prove Eq. 7.2, we need to focus only on the interaction between A and its oracle. Note that when D' returns $\perp$ (error) at the end of a conversation, then SIM will also return $\perp$ since SIM carries out the same verification procedure as D'. Thus, D' and SIM can only differ in their behavior when the interaction is a valid one. Observe again that if SIM returns a message $m \neq \perp$ then D' should output the same message. This

is true because if SIM outputs m, then $\mathcal{K}$ must have found q such that $M_{pk}(q) = c_1$ and $m = y \oplus c_2$. Since M_{pk} is a trapdoor permutation, D' computes the same q and consequently, the same message. Furthermore, if D' outputs a valid plaintext while SIM returns $\perp$, then by collision resistance, $\mathcal{K}$ could not have computed a consistent preimage, $q' \neq q$ (except with negligible probability, which is accounted for by μ in Eq. 7.2). Otherwise, a collision (e.g., q, q', c_1) occurs between q', as computed by $\mathcal{K}$, and $q = M_{pk}^{(-1)}(c_1)$. Therefore, the only noticeable way in which D' and SIM can differ is when $\mathcal{K}$ fails in finding a preimage. This event happens with probability at most $\frac{1}{6p_A^2} + \mu \leq \frac{1}{3p_A^2}$ per decryption query. Since there are at most p_A decryption queries in the game, then by the union bound such a bad event happens with probability at most $\frac{p_A}{3p_A^2} \leq \frac{1}{3p_A} \leq \frac{1}{3p}$.[2] This proves Eq. 7.2.

Putting all the pieces together, we have for all $n \in \mathbb{N}'$:

$$Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s) \leftarrow B_1(pk'),\ c \leftarrow< E'(m_0, pk'), D'(sk') >,$$

$$b \leftarrow B_2(s, c) : b = 1]\ -$$

$$Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s) \leftarrow B_1(pk'),\ c \leftarrow< E'(m_1, pk'), D'(sk') >,$$

$$b \leftarrow B_2(s, c) : b = 1]$$

$$= Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s) \leftarrow A_1^{SIM(pk')}(pk'),\ c \leftarrow< E'(m_0, pk'), D'(sk') >,$$

$$b \leftarrow A_2^{SIM(pk')}(s, c) : b = 1]-$$

$$Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s) \leftarrow A_1^{SIM(pk')}(pk'),\ c \leftarrow< E'(m_1, pk'), D'(sk') >,$$

$$b \leftarrow A_2^{SIM(pk')}(s, c) : b = 1]$$

$$\geq Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s) \leftarrow A_1^{D'(sk')}(pk'),\ c \leftarrow< E'(m_0, pk'), D'(sk') >,$$

$$b \leftarrow A_2^{D'(sk')}(s, c) : b = 1]-$$

$$Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s) \leftarrow A_1^{D'(sk')}(pk'),\ c \leftarrow< E'(m_1, pk'), D'(sk') >,$$

$$b \leftarrow A_2^{D'(sk')}(s, c) : b = 1] - \frac{2}{3p(n)} - \mu(n) \tag{7.3}$$

[2] Recall that $p_A(n) \geq p(n),\ \forall n$.

$$\geq \frac{1}{3p(n)} - \mu(n), \tag{7.4}$$

where Eq. 7.3 holds due to Eq. 7.2 , and Eq. 7.4 holds by Eq. 7.1. A contradiction with IND-CPA. Therefore, this construction is IND-CCA2. □

7.3 Towards Instantiation of General Encryption Schemes

In this section, we investigate a general question regarding encryption schemes in the RO model. In particular, what does security of such schemes tells us about the security of the "instantiated" ones? As elaborated in [CGH98], security of encryption schemes in the RO model does not guarantee, in general, security in the standard model, i.e., when each Random Oracle query is replaced by a call to a function fixed at the beginning of the protocol. This is so because no efficient function satisfies both unpredictability and consistency, two properties of Random Oracles. Unpredictability means that answers to different queries are uniform and independent while consistency means that answers to the *same* queries are the same. To get around this impossibility result, we propose a new construct that satisfies both properties. At a high level, this construct assumes that the encryption algorithm, E, and decryption algorithm, D share a secret. Then, answers to *new* Random Oracle queries are chosen uniformly and independently (thus, achieving unpredictability), and transmitted to the other party using this secret, e.g., by using a symmetric encryption scheme based on the secret, to guarantee consistency.

Before we discuss our instantiation in more detail, a few words are due about our assumption that E and D share a secret. This requirement may be implemented in several ways. Our solution is to assume that the *first* Random Oracle query made by both parties is the same but can not be efficiently retrieved from the ciphertext without the decryption key. If so, then the secret can be designated as the first Random Oracle query itself. Formally, our requirements on the original encryption scheme are:

Definition 7.3.1 (First-Query Hiding Encryption Schemes). *An encryption scheme in the RO model, $P = (G, E, D)$, is called **first-query hiding** if it satisfies the following two conditions:*

- *A ciphertext, c, reveals the number of oracle queries that E makes to compute c. Denote this number by d_c.*

- ***If*** *E* ***makes a Random Oracle query****, then the first such query, q_1, satisfies the following three conditions:*
  - *q_1 is taken from a well-spread distribution.*
  - *q_1 is also the first query that D makes.*
  - *For any message, m, and any PPT, $\mathcal{K}$:*

$$Pr[(pk, sk) \leftarrow G(1^n),\ c \leftarrow E(m, pk),\ q \leftarrow \mathcal{K}(c) : q = q_1] \leq \mu(n),$$

where q_1 is the first query made by E while computing c.

We emphasize that first-query hiding encryption schemes need not use the Random Oracle. Thus, this class of encryption schemes include all schemes in the standard model.

We proceed as follows. We show how to convert any first-query hiding IND-CPA encryption scheme in the RO model to a CCA2 encryption scheme. This result combines two steps in one. Specifically, it consists of converting any first-query hiding IND-CPA encryption scheme to an IND-CPA encryption scheme in the standard model and then converting the latter to a CCA2 encryption scheme. After we present this result in the interactive model, we explain how this result can be applied in the noninteractive setting with the help of noninteractively-extractable POW functions.

7.3.1 Interactive Instantiation

Our starting point is a semantically secure encryption scheme in the RO model, denoted by $P = (G, E, D)$, and an (interactively) extractable POW family ensemble, $\mathbb{H}$, satisfying strong pseudorandomness (Definition 4.3.5). Given these two primitives, we construct a 3-round encryption scheme, $P' = (G', E', D')$, secure against chosen message attack in the standard model. At a high level, E' runs E, with E' providing uniform and independent answers to Random Oracle queries made by E. When E halts with a ciphertext, E' verifies to D' that it knows the corresponding message and the secret random coins of E. This is done using the extractability property of $\mathbb{H}$. Also, it sends the ciphertext computed by E as well as all the Random Oracle query and answer pairs. D' uses D to decrypt with the query/answer pairs sent by E' acting as a Random Oracle (this oracle is defined formally in Algorithm 7.3.2). When D finishes, D' verifies that

the interaction is consistent (as in Algorithm 7.3.3) before returning the plaintext.

In more detail, to encrypt a message, m, E' chooses private random coins for E, denoted by r^E, computes the encryption of m under E using r^E as well as the encryption of r^E. As mentioned earlier, answers to Random Oracles queries are chosen uniformly and independently by E'. Then, E' engages in a 3-round interaction with D' to prove knowledge of m, r^E. In addition, E' sends both ciphertexts (of m and r^E) as well as an encryption of Random Oracle query/answer pairs in the 3^{rd} round. Each oracle query/answer (q_i, u_i) is encrypted as:

$$Enc_{q_1}(q_i, u_i) = H_k((q_1, q_i), r_0^{q_i}), H_k((q_1, q_i, u_i^1), r_1^{q_i}.), ..., H_k((q_1, q_i, u_i^n), r_n^{q_i}),$$

where u_i^j is the jth bit of u_i. Notice that it is possible to compute u_i given $Enc_{q_1}(q_i, u_i)$,q_1, and q_i. Specifically, $u_i^j = 1$ if and only if $V_{\mathbb{H}}((q_1, q_i, 1), H_k((q_1, q_i, u_i^j), r_j^{q_i})) = 1$.

Formally, let $P' = (G', E', D')$, with G' defined in Algorithm 7.3.1 and E', D' defined in Figure 7.2. We assume without loss of generality that both E and D do not repeat a Random Oracle query, e.g., E and D remember answers to previous queries. It can be shown that P' satisfies completeness. In Theorem 7.3.1, we state that it is IND-CCA2.

input: 1^n

1 $(pk_1, sk_1) \leftarrow G(1^n)$;

2 $(pk_2, sk_2) \leftarrow G(1^n)$;

3 $k \leftarrow K_n$;

4 $pk' \triangleq (pk_1, pk_2, k)$;

5 $sk' \triangleq (sk_1, sk_2, k)$;

6 **return** (pk', sk');

Algorithm 7.3.1: G'

input: $k, y_0^{q_1}, ..., y_n^{q_1}, ..., y_0^{q_d}, ..., y_n^{q_d}, q_1, q'$

if $\exists i,\ V_{\mathbb{H}}((q_1, q'), y_0^{q_i}) = 1$ **and** $\forall j < i,\ V_{\mathbb{H}}((q_1, q'), y_0^{q_j}) = 0$ **then**
    $u = V_{\mathbb{H}}((q_1, q_i, 1), y_1^{q_i}), ..., V_{\mathbb{H}}((q_1, q_i, 1), y_n^{q_i})$;
    return u;
else
    $u \leftarrow U_n$;
    return u;
end

Algorithm 7.3.2: O'

input: $pk', y_0, ..., y_n, r_1, ..., r_n, y_0^{q_1}, ..., y_n^{q_1}, ..., y_0^{q_d}, ..., y_n^{q_d}, c_1, c_2, m', r'$

if $V_{\mathbb{H}}((m', r'), y_0) = 1$ **and** $\forall i \geq 1,\ H_k((m', r'), r_i) = y_i$ **and** $d = d_{c_1} + d_{c_2}$ **and** $E^{O'(k, y_0^{q_1}, ..., y_n^{q_1}, ..., y_0^{q_{d_{c_1}}}, ..., y_n^{q_{d_{c_1}}}, q_1, .)}(m', pk_1, r') = c_1$ **then**
    for $i = 1$ **to** d_{c_1} **do**
        let q_i' be the ith query made by $E(m', pk_1, r')$;
        if $V_{\mathbb{H}}((q_1', q_i'), y_0^{q_i}) \neq 1$ **or** $\exists j \geq 1, V_{\mathbb{H}}((q_1', q_i', 0), y_j^{q_i}) = V_{\mathbb{H}}((q_1', q_i', 1), y_j^{q_i})$ **then**
            return 0 ;
    end
    return 1 ;
else
    return 0;
end

Algorithm 7.3.3: Ver

Theorem 7.3.1. *Let $P = (G, E, D)$ be any first-query hiding encryption scheme that is IND-CPA in the RO model, and $\mathbb{H}$ be any extractable (as in Definition 4.2.6) family ensemble that satisfies strong pseudorandomness with auxiliary information (as in Definition 4.3.5), collision resistance (as in Definition 2.5.2) and has public randomness. Then, P', the corresponding protocol in Figure 7.2, is IND-CCA2 (as in Definition*

$E'(m, pk')$ | | $D'(sk')$

$r^E \leftarrow R_n^E$
$c_1 = E(m, pk_1, r^E)$
$c_2 \leftarrow E(r^E, pk_2)$
$d = d_{c_1} + d_{c_2}$
$u_1, ..., u_d \leftarrow U_n, ..., U_n$
use $u_1, ..., u_d$ as RO answers to $q_1, ..., q_d$

$y_0^{q_1}, y_1^{q_1}, ..., y_n^{q_1} \leftarrow H_k(q_1, q_1, .), H_k(q_1, q_1, u_1^1, .), ..., H_k(q_1, q_1, u_1^n, .)$
$\vdots$
$y_0^{q_d}, y_1^{q_d}, ..., y_n^{q_d} \leftarrow H_k(q_1, q_d, .), H_k(q_1, q_d, u_d^1, .), ..., H_k(q_1, q_d, u_d^n, .)$

$y_0 \leftarrow H_k((m, r^E))$

$\xrightarrow{y_0}$

$r_1, ..., r_n \leftarrow R_n, ..., R_n$

$\xleftarrow{r_1, ..., r_n}$

$y_1 = H_k((m, r^E), r_1), ..., y_n = H_k((m, r^E), r_n)$

$\xrightarrow{y_1, ..., y_n, y_0^{q_1}, ..., y_n^{q_d}, c_1, c_2}$

$m' = D^{O'(k, y_0^{q_1}, ..., y_n^{q_1}, ..., y_0^{q_{d_{c_1}}}, ..., y_n^{q_{d_{c_1}}}, q_1, .)}(c_1, sk_1)$
$r' = D^{O'(k, y_0^{q_{d_{c_1}+1}}, ..., y_n^{q_{d_{c_1}+1}}, ..., y_0^{q_{d_{c_2}}}, ..., y_n^{q_{d_{c_2}}}, q_1, .)}(c_2, sk_2)$
//Recall D asks q_1 as the first query
if $Ver(pk', y_0, ..., y_n, r_1, ..., r_n, y_0^{q_1}, ..., y_n^{q_1}, ..., y_0^{q_d}, ..., y_n^{q_d}, c_1, c_2, m', r') = 1$:
output m'
otherwise, output $\perp$

Figure 7.2: Interactive Instantiation of First-query Hiding Encryption Schemes

7.2.2).

Proof. The proof is by contradiction. We suppose P' is not IND-CCA2, then we use a reducibility argument to conclude that P is not semantically secure.

In more detail, if A is an adversary that defeats P', we construct another adversary, B, that runs A to defeat P. Two major issues emerge when B runs A, namely decryption queries and the challenge ciphertext. Recall that B is a CPA adversary and as such does not have access to a decryption oracle while A does. So, when B runs A it has to answer A's decryption queries on its own. Moreover, B receives a challenge ciphertext under P while A expects a proper challenge ciphertext under P'. Thus, B needs to convert the former to the latter.

To resolve the first issue, B simulates a decryption oracle by using a knowledge extractor to find the plaintext. This simulation is formally defined in Algorithm 7.3.6. Regarding the second issue, B extends its challenge ciphertext to a new string that is indistinguishable from a valid ciphertext under P' and then runs A on it. This indistinguishability argument uses the assumption that $\mathbb{H}$ is perfectly one-way. The conversion is formally defined in Algorithm 7.3.7.

The proof proceeds as follows. First, we define B. Second, we use an indistinguishability argument to show that the noticeable advantage A has when playing the CCA2 game as in Definition 7.2.2 translates to a noticeable advantage when A runs in the simulated world of B. Thus, B has a noticeable advantage in defeating P.

Analysis

Formally, suppose, for the purpose of contradiction, that scheme P' does not satisfy definition 7.2.2. Then, there exists a PPT pair $A = (A_1, A_2)$, an infinite set of security parameters $\mathbb{N}' \subseteq \mathbb{N}$, and a polynomial p, such that:

$$|Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s') \leftarrow A_1^{D'(sk')}(pk'),\ c' \leftarrow< E'(m_0, pk'), D'(sk') >,$$

$$b \leftarrow A_2^{D'(sk')}(s', c') : b = 1] \ -$$

$$Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s') \leftarrow A_1^{D'(sk')}(pk'),\ c' \leftarrow< E'(m_1, pk'), D'(sk') >,$$

$$b \leftarrow A_2^{D'(sk')}(s', c') : b = 1]| > \frac{1}{p(n)}. \tag{7.5}$$

input: pk_1

$(pk_2, sk_2) \leftarrow G(1^n)$;
$k \leftarrow K_n$;
$pk' = (pk_1, pk_2, k)$;
$(m_0, m_1, s') \leftarrow A_1^{SIM(pk', sk_2)}(pk')$;
return $m_0, m_1, s = (s', pk', sk_2)$;

Algorithm 7.3.4: B_1

input: $c_1, s = (s', pk', sk_2)$

$c' \leftarrow$`Extend` (c_1, s);
$b \leftarrow A_2^{SIM(pk', sk_2)}(c', s')$;
return b;

Algorithm 7.3.5: B_2

Let p_A be a polynomial bounding the running time of A. Then, A can ask at most p_A decryption queries. Without loss of generality, we assume that $p_A(n) \geq p(n), \forall n$. Let $\mathcal{K}$ be a blackbox extractor that succeeds with probability at least $1 - \frac{1}{6p_A^2} - \mu$ (as in Definition 4.2.6). We use A to construct a PPT pair, $B = (B_1, B_2)$, that attacks P. B is formally defined in Algorithms 7.3.4 and 7.3.5.

input : pk', sk_2

interaction: with an external PPT, A

$(m', r') \leftarrow \mathcal{K}^A(k)$;
receive y_0;
$r_1, ..., r_n \leftarrow R_n, ..., R_n$;
send $r_1, ..., r_n$;
receive $y_1, ..., y_n, y_0^{q_1}, ..., y_n^{q_d}, c_1, c_2$;
let q_1' be the first query $E(m', pk_1, r')$ makes;
if $Ver(pk', y_0, ..., y_n, r_1, ..., r_n, y_0^{q_1}, ..., y_n^{q_1}, ..., y_0^{q_d}, ..., y_n^{q_d}, c_1, c_2, m', r') = 1$ **and** $D^{O'(k, y_0^{q_1}, ..., y_n^{q_1}, ..., y_0^{q_{dc_1}}, ..., y_n^{q_{dc_1}}, q_1, .)}(c_2, sk_2) = r'$ **then**
    return m';
else
    return $\perp$;
end

Algorithm 7.3.6: SIM

input: c_1, s

1 $r_1^E \leftarrow R_n^E$;

2 $c_2 \leftarrow E(r_1^E, pk_2)$;

3 $y_0, ..., y_n \leftarrow U_n, ..., U_n$;

4 $y_0^{q_1}, ..., y_n^{q_1}, ..., y_0^{q_d}, ..., y_n^{q_d} \leftarrow U_n, ..., U_n$;

5 $c' = y_0, ..., y_n, y_0^{q_1}, ..., y_n^{q_1}, ..., y_0^{q_d}, ..., y_n^{q_d}, c_1, c_2$;

6 **return** c';

Algorithm 7.3.7: *Extend*

We argue that the behavior of A does not differ much when playing the real CCA2 game from the simulated game. In particular, we show that if we replace the decryption oracle, D', by SIM, A's output remains the same except with probability $\frac{1}{3p}$. Moreover, when a valid ciphertext, c', is replaced by $Extend(s, c_1)$, A's output differ only with negligible probability.

Formally, we need to show that the following two equations hold for all sufficiently large $n \in \mathbb{N}'$ and any $i \in \{0, 1\}$:

$$|Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s') \leftarrow A_1^{SIM(pk', sk_2)}(pk'),\ c' \leftarrow < E'(m_i, pk'), D'(sk') >$$

$$b \leftarrow A_2^{SIM(pk', sk_2)}(s', c') : b = 1] -$$

$$Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s') \leftarrow A_1^{D'(sk')}(pk'),\ c' \leftarrow < E'(m_i, pk'), D'(sk') >$$

$$b \leftarrow A_2^{D'(sk')}(s', c') : b = 1]| \leq \frac{1}{3p(n)} + \mu(n). \tag{7.6}$$

$$|Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s') \leftarrow A_1^{SIM(pk', sk_2)}(pk'),\ c \leftarrow E^O(m_i, pk_1),$$

$$c' \leftarrow Extend(c, s),\ b \leftarrow A_2^{SIM(pk', sk_2)}(s', c') : b = 1] -$$

$$Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s') \leftarrow A_1^{SIM(pk', sk_2)}(pk'),\ c' \leftarrow < E'(m_i, pk'), D'(sk') >$$

$$b \leftarrow A_2^{D'(sk')}(s', c') : b = 1]| \leq \mu(n). \tag{7.7}$$

To prove Eq. 7.6, we need to focus only on the interaction between A and its oracle. Note that if D' returns $\perp$, then SIM also returns $\perp$. Moreover, if SIM returns a valid plaintext, m', then by collision resistance, D' returns the same message, m'. Therefore, the only way D' and SIM may differ is when D' returns m' but SIM outputs $\perp$. This happens when $\mathcal{K}^A$ fails to find a valid preimage, i.e., it happens with probability at most $\frac{1}{6p_A^2} + \mu \leq \frac{1}{3p_A^2}$. Therefore, by the union bound, the probability that D' and SIM differ in any interaction with A is at most $\frac{p_A}{3p_A^2} \leq \frac{1}{3p_A} \leq \frac{1}{3p}$.

Eq. 7.7 is true because P is first-query hiding and semantically secure and $\mathbb{H}$ satisfies Definition 4.3.5. In more detail, semantic security implies that $c_1 = E(m_i, pk_1, r^E), c_2 \leftarrow E(r^E, pk_2)$ is indistinguishable from $c_1, E(U_n, pk_2)$. Thus, c_1, c_2 is uninvertible in both q_1 and (m_i, r^E), because $c_1, E(U_n, pk_2)$ is uninvertible in both q_1 (by first-query hiding) and (m_i, r^E) (we assume, without loss of generality that R_n^E is a well-spread distribution on strings of length at least n). In other words, c_1, c_2 can be regarded as auxiliary information about either q_1 or (m_i, r^E). Thus, by Definition 4.3.5, $y_0^{q_1}, ..., y_n^{q_1}, ..., y_0^{q_d}, ..., y_n^{q_d}, c_1, c_2$ is indistinguishable from $U_{dl(n)+dnl(2n+1)}, c_1, c_2$. Consequently, $y_0^{q_1}, ..., y_n^{q_1}, ..., y_0^{q_d}, ..., y_n^{q_d}$, c_1, c_2 is uninvertible in (m_i, r^E). Moreover, using Definition 4.3.5 again, $y_0, ..., y_n, y_0^{q_1}, ...,$ $y_n^{q_1}, ..., y_0^{q_d}, ..., y_n^{q_d}, c_1, c_2$ is indistinguishable from $U_{(n+1)l(n)}, y_0^{q_1}, ..., y_n^{q_1}, ..., y_0^{q_d}, ..., y_n^{q_d}, c_1, c_2$ which, as argued earlier, is indistinguishable from $U_{(n+1)l(n)}, U_{dl(n)+dnl(2n+1)}, c_1, E(U_n, pk_2)$. Eq. 7.7 follows.

Putting all the pieces together, we have for all $n \in \mathbb{N}'$:

$$|Pr[(pk, sk) \leftarrow G(1^n),\ (m_0, m_1, s) \leftarrow B_1(pk),\ c \leftarrow E^O(m_0, pk),\ b \leftarrow B_2(s, c) : b = 1]\ -$$

$$Pr[(pk, sk) \leftarrow G(1^n),\ (m_0, m_1, s) \leftarrow B_1(pk),\ c \leftarrow E^O(m_1, pk),\ b \leftarrow B_2(s, c) : b = 1]|$$

$$= |Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s') \leftarrow A_1^{SIM(pk', sk_2)}(pk'),\ c \leftarrow E^O(m_0, pk_1),$$

$$c' \leftarrow Extend(s, c),\ b \leftarrow A_2^{SIM(pk', sk_2)}(s', c') : b = 1]-$$

$$Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s') \leftarrow A_1^{SIM(pk', sk_2)}(pk'),\ c \leftarrow E^O(m_1, pk_1),$$

$$c' \leftarrow Extend(s, c),\ b \leftarrow A_2^{SIM(pk', sk_2)}(s', c') : b = 1]|$$

$$\geq |Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s') \leftarrow A_1^{D'(sk')}(pk'),\ c' \leftarrow< E'(m_0, pk'), D'(sk') >,$$

$$b \leftarrow A_2'^{D'(sk')}(s', c') : b = 1]-$$

$$Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s') \leftarrow A_1^{D'(sk')}(pk'),\ c' \leftarrow< E'(m_1, pk'), D'(sk') >,$$

$$b \leftarrow A_2^{D'(sk')}(s', c') : b = 1]| - \frac{2}{3p(n)} - \mu(n) \tag{7.8}$$

$$\geq \frac{1}{3p(n)} - \mu(n), \tag{7.9}$$

where Eq. 7.8 holds due to Eq. 7.6 and Eq. 7.7, and Eq. 7.9 holds due to Eq. 7.5, a contradiction with the assumption that P is IND-CPA. □

Remark 7.3.1. *Observe that in the proof of Theorem 7.3.1, B did not access the RO. Thus, Theorem 7.3.1 still holds if P is not IND-CPA (as in definition 2.7.1) but is so against adversaries without access to the Random Oracle. For instance, if M_{pk} is a trapdoor permutation, $M_{pk}(r_1), r_2, O(r_2) \oplus m \leftarrow E^O(m, pk)$ is not IND-CPA but P', as defined in Figure 7.2, is CCA2-secure, provided that the first Random Oracle query of both E and D is r_1. This is true because Random Oracle answers can not be recovered in P' without knowledge of r_1.*

7.3.2 Noninteractive Instantiation

7.3.2.1 IND-CPA Instantiation

We remark that P' (Figure 7.2) can be modified so that it becomes *noninteractively* IND-CPA in the standard model. For this result, we have the same assumptions on P while we assume that $\mathbb{H}$ is only a (strong pseudorandom) POW family ensemble. We emphasize that we do not assume any extractability property on $\mathbb{H}$. The formal construction is as follows.

Construction 7.3.1. *Let $P = (G, E, D)$ be any first-query hiding encryption scheme in the RO model and $P' = (G', E', D')$ be the following encryption scheme in the standard model.*

- $G'(1^n) = (pk, sk, k)$, *where* $(pk, sk) \leftarrow G(1^n)$ *and* $k \leftarrow K_n$.

- $E'(m, pk, k) = c_0, c_1$

$$c_0 \triangleq Enc_{q_1}(q_1, u_1), ..., Enc_{q_1}(q_{d_{c_1}}, u_{d_{c_1}}),$$
$$c_1 \triangleq E^{O'(k, Enc_{q_1}(q_1,u_1),...,Enc_{q_1}(q_{d_{c_1}},u_{d_{c_1}}),q_1,.)}(m, pk_1, r_E),$$

where r_E, r'_E, *and* $u_1, ..., u_d$ *are uniform,* q_1 *is the first query that* $E^{O'}(m, pk_1, r_E)$ *makes,* O' *is defined in Algorithm 7.3.2, and* $Enc_{q_1}(q_i, u_i) = H_k(q_1, q_i, r_0), H_k(q_1, q_i, u_i^1, r_1), ..., H_k(q_1, q_i, u_i^n, r_n)$ *(where* $r_0, ..., r_n$ *are uniform).*

- $D'(c = (c_0, c_1), sk, k) = D^{O'(k, c_0, q_1, .)}(c_1, sk)$

Theorem 7.3.2. *Let* $P = (G, E, D)$ *be any first-query hiding encryption scheme that is IND-CPA in the RO model (as in Definition 2.7.1), and* $\mathbb{H}$ *be any family ensemble that satisfies strong pseudorandomness with auxiliary information (as in Definition 4.3.5), and collision resistance (as in Definition 2.5.2). Then, Construction 7.3.1 is IND-CPA (as in Definition 2.7.1).*

Proof. The proof of this theorem is very similar to the proof of Eq. 7.7 (in the proof of Theorem 7.3.1). By perfect one-wayness with auxiliary information and the fact that c_1 is one-way in q_1 (first-query hiding), we know that c_0, c_1 is indistinguishable from $U_{|c_0|}, c_1$. Therefore, any noticeable advantage that an adversary has against c_0, c_1, it also has it against $U_{|c_0|}, c_1$. However, by semantic security of P, it is not possible to tell whether $U_{|c_0|}, c_1$ (and consequently c_0, c_1) is an encryption of m_0 or m_1. Note that the fact that E is simulated with O' does not alter semantic security of P because O' behaves exactly like a Random Oracle from the perspective of E.

We need collision resistance to show completeness. Specifically, collision resistance implies that $u_1, ..., u_{d_{c_1}}$ that O' recovers are the same as those used by E' and consequently D and E use the same Random Oracle. Thus, completeness follows from completeness of the underlying scheme. □

7.3.2.2 IND-CCA2 Instantiation

The following modification to Construction 7.3.1 is a CCA2-secure instantiation of any first-query hiding encryption scheme, provided that $\mathbb{H}$ is an extractable POW family ensemble. We emphasize that the adversary is restricted to asking a constant number of decryption queries.

Construction 7.3.2. *Let $P = (G, E, D)$ be any first-query hiding encryption scheme in the RO model and $P' = (G', E', D')$ be the following encryption scheme in the standard model.*

- $G'(1^n) = (pk_1, pk_2, sk_1, sk_2, k)$, *where* $(pk_1, sk_1), (pk_2, sk_2) \leftarrow G(1^n)$ *and* $k \leftarrow K_n$.
- $E'(m, pk_1, pk_2, k) = H_k((m, r^E, c_0, c_2), r_{\mathbb{H}}), c_0, c_1, c_2$

$$c_0 \triangleq Enc_{q_1}(q_1, u_1), ..., Enc_{q_1}(q_d, u_d),$$
$$c_1 \triangleq E^{O'(k, Enc_{q_1}(q_1,u_1),...,Enc_{q_1}(q_d,u_d), q_1, .)}(m, pk_1, r_E),$$
$$c_2 \triangleq E^{O'(k, Enc_{q_1}(q_1,u_1),...,Enc_{q_1}(q_d,u_d), q_1, .)}(r_E, pk_2, r'_E),$$

  where r_E, r'_E, $r_{\mathbb{H}}$, and $u_1, ..., u_d$ are uniform, q_1 is the first query that $E^{O'}(m, pk_1, r_E)$ makes, O' is defined in Algorithm 7.3.2, and $Enc_{q_1}(q_i, u_i) = H_k(q_1, q_i, r_0), H_k(q_1, q_i, u_i^1, r_1), ..., H_k(q_1, q_i, u_i^n, r_n)$ (where $r_0, ..., r_n$ are uniform).

- $D'((y, c_0, c_1, c_2), pk_1, sk_1, pk_2, sk_2, k)$. *Let q_1 be the first query that $D^{O'(k,c_0,.,.)}(c_1, sk_1)$ makes, $D^{O'(k,c_0,q_1,.)}(c_1, sk_1) = m$, and $D^{O'(k,c_0,q_1,.)}(c_2, sk_2) = r^E$. Then D' outputs m if $Ver2(y, c_0, c_1, c_2, m, r^E, q_1, pk_1, pk_2, k) = 1$ ($Ver2$ is defined in Algorithm 7.3.8). Otherwise, D' returns $\perp$.*

input: $y, c_0, c_1, c_2, m, r^E, q_1, pk_1, pk_2, k$

```
1  let q'_1 be the first query and d_{c_1} be the number of queries that
   E^{O'(k,c_0,q_1,.)}(m, pk_1, r^E) makes.;
2  E^{O'(k,c_0,q_1,.)}(m, pk_1, r^E) = c'_1;
3  if q_1 ≠ q'_1 or c'_1 ≠ c_1 or V_H((m, r^E, c_0, c_2), y) ≠ 1 then
4      return 0 ;
5  else
6      interpret y as y_0^{q_1}, ..., y_n^{q_1}, ..., y_0^{q_{d_{c_1}}}, ..., y_n^{q_{d_{c_1}}}, ..., y_n^{q_d};
7      for i = 1 to d_{c_1} do
8          let q'_i be the ith query made by E^{O'(k,c_0,q_1,.)}(m, pk_1, r^E);
9          if V_H((q'_1, q'_i), y_0^{q_i}) ≠ 1 or ∃j ≥ 1, V_H((q'_1, q'_i, 0), y_j^{q_i}) = V_H((q'_1, q'_i, 1), y_j^{q_i})
           then
10             return 0 ;
11     end
12     return 1 ;
13 end
```

Algorithm 7.3.8: Ver2

Theorem 7.3.3. *Let $P = (G, E, D)$ be any first-query hiding encryption scheme that is IND-CPA in the RO model (as in Definition 2.7.1), and $\mathbb{H}$ be any family ensem-*

ble that satisfies strong pseudorandomness with auxiliary information (as in Definition 4.3.5), extraction with auxiliary input (as in Definition 3.2.5), and collision resistance (as in Definition 2.5.2). Then, Construction 7.3.2 is IND-CCA2 (as in Definition 2.7.2), against a constant number of decryption queries.

Proof. The proof follows very similar lines to the proof of Theorem 7.3.1. Thus, the formal proof is recreated here without high-level description.

Suppose, for the purpose of contradiction, that Construction 7.3.2 does not satisfy definition 2.7.2. Then, there exists a PPT pair $A = (A_1, A_2)$, an infinite set of security parameters $\mathbb{N}' \subseteq \mathbb{N}$, and a polynomial p, such that:

$$|Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s') \leftarrow A_1^{D'(sk')}(pk'),\ c' \leftarrow E'(m_0, pk'),$$

$$b \leftarrow A_2^{D'(sk')}(s', c') : b = 1] \ -$$

$$Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s') \leftarrow A_1^{D'(sk')}(pk'),\ c' \leftarrow E'(m_1, pk'),$$

$$b \leftarrow A_2^{D'(sk')}(s', c') : b = 1]| > \frac{1}{p(n)}. \tag{7.10}$$

Let $\mathcal{K}$ be a preimage extractor for all of the constant number of queries that A makes, as described in the proof of Theorem 7.2.1. Define a new PPT pair, $B = (B_1, B_2)$ that uses A to break semantic security of the underlying scheme, P, as in Algorithms 7.3.4 and 7.3.5 (with *Extend* and SIM defined in Algorithms 7.3.9 and 7.3.10).

input: c_1, s
1 $r^E \leftarrow R_n^E$;
2 $c_2 \leftarrow E(r^E, pk_2)$;
3 $c_0 = y_0^{q_1}, ..., y_n^{q_1}, ..., y_0^{q_d}, ..., y_n^{q_d} \leftarrow U_n, ..., U_n$;
4 $y \leftarrow U_{|H_k(m_b, r^E, c_0, c_2)|}$;
5 $c' = y, c_0, c_1, c_2$;
6 **return** c';

Algorithm 7.3.9: *Extend*

We argue that the advantage that A has when interacting with D' remains when interacting with SIM. Formally, for sufficiently large $n \in \mathbb{N}'$ and any $i \in \{0, 1\}$:

$$|Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s') \leftarrow A_1^{SIM(pk', sk_2)}(pk'),\ c' \leftarrow E'^O(m_i, pk'),$$

input : $pk_1, pk_2, k, sk_2, y, c_0, c_1, c_2$

1 $(m', r'^E, c'_0, c'_2) \leftarrow \mathcal{K}_A(k, r_A)$;
2 let q'_1 be the first query $E(m', pk_1, r')$ makes;
3 **if** $Ver2(y, c_0, c_1, c_2, m', r'^E, q'_1, pk_1, pk_2, k) = 1$ **and** $D^{O'(k,c_0,q_1,.)}(c_2, sk_2) = r'^E$ **and** $(c'_0, c'_2) = (c_0, c_2)$ **then**
4 **return** m';
5 **else**
6 **return** $\perp$;
7 **end**

Algorithm 7.3.10: SIM

$$b \leftarrow A_2^{SIM(pk', sk_2)}(s', c') : b = 1]-$$

$$Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s') \leftarrow A_1^{D'(sk')}(pk'),\ c' \leftarrow E'^O(m_i, pk'),$$

$$b \leftarrow A_2^{D'(sk')}(s', c') : b = 1]| \leq \mu(n). \tag{7.11}$$

To prove Eq. 7.11, we need to focus only on the interaction between A and its oracle. Observe that the output of D' can be different from that of SIM only if there is a difference on m, r^E, q_1. First, note that SIM computes q'_1 as the first Random Oracle query made by E while D' computes q_1 as the first Random Oracle query made by D. By the first-query hiding property, $q'_1 = q_1$.

Moreover, if D' outputs $m \neq \perp$, then by construction, we know that D' runs $Ver2$ on m, r^E (among other inputs) and $Ver2$ returns 1. Thus, $V_{\mathbb{H}}((m, r^E, c_0, c_2), y) = 1$ (this is verified by $Ver2$). Since y has a valid preimage, $\mathcal{K}_A$ returns m', r'^E, c'_0, c'_2 such that $V_{\mathbb{H}}((m', r'^E, c'_0, c'_2), y) = 1$ (except with negligible error). By collision resistance, $(m, r^E, c_0, c_2) = (m', r'^E, c'_0, c'_2)$. Thus, $m, r^E, q_1 = m', r'^E, q'_1$. Since $r^E, c'_0, c'_2 = r'^E, c_0, c_2$ and r^E is the plaintext of c_2, then the second and third conditions on line 3 of SIM is valid. Consequently both D' and SIM have the same output, m.

In addition, if SIM returns $m' \neq \perp$, then by construction, $\mathcal{K}_A$ recovers m', r'^E, c'_0, c'_2 such that $Ver2$ returns 1 on m', r'^E, q'_1 (among other inputs), $c'_0, c'_2 = c_0, c_2$, m' is the plaintext of c_1, and r'^E is the plaintext of c_2. By correctness, $D^{O'(k,c_0,q_1,.)}(c_1, sk_1) = m'$ and $D^{O'(k,c_0,q_1,.)}(c_2, sk_2) = r'^E$. Thus, D' recovers $m, r^E, q_1 = m', r'^E, q'_1$, and outputs m' because $Ver2$ accepts.

Next, we show that we do not loose the advantage of A if we run it on $Extend(c_1)$

instead of a valid ciphertext. Formally, for any $i \in \{0,1\}$ and sufficiently large $n \in \mathbb{N}'$:

$$|Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s') \leftarrow A_1^{SIM(pk', sk_2)}(pk'),\ c \leftarrow E^O(m_i, pk_1),$$

$$c' \leftarrow Extend(c, s),\ b \leftarrow A_2^{SIM(pk', sk_2)}(s', c') : b = 1] -$$

$$Pr[(pk', sk') \leftarrow G'(1^n),\ (m_0, m_1, s') \leftarrow A_1^{SIM(pk', sk_2)}(pk'),\ c' \leftarrow E'(m_i, pk')$$

$$b \leftarrow A_2^{SIM(pk', sk_2)}(s', c') : b = 1]| \leq \mu(n). \tag{7.12}$$

Note that semantic security of P is not violated if we replace the Random Oracle, O, with $O'(k, c_0, q_1, .)$ because the answers of O' are uniform and independent (like O). For any $i \in \{0,1\}$, let:

$$c_1^{m_i} \triangleq E^{O'(k, Enc_{q_1}(q_1,u_1),\ldots,Enc_{q_1}(q_d,u_d), q_1, .)}(m_i, pk_1, r_E),$$

$$c_2^{m_i} \triangleq E^{O'(k, Enc_{q_1}(q_1,u_1),\ldots,Enc_{q_1}(q_d,u_d), q_1, .)}(r_E, pk_2, r'_E)$$

Observe that semantic security implies that r^E is well-spread. Otherwise, a *nonuniform* adversary, A, can distinguish an encryption of m_0 from an encryption of m_1. A guesses r^E with nonnegligible advantage (or the particular r^E with the nonnegligible probability weight can be given to A as an advice string). Then, A recomputes $c' = E(m_0, pk, r^E)$ and $c'' = E(m_1, pk, r^E)$. Note that by correctness of the encryption scheme, $c' \neq c''$. Then, A compares c' and c'' to the actual ciphertext. If there is a match with m_i, A outputs i. Otherwise, output a uniform bit. It follows that A has a nonnegligible advantage against semantic security.

By semantic security and the fact that r^E is drawn from a well-spread distribution, for any i, $(c_1^{m_i}, c_2^{m_i})$ is computationally indistinguishable from

$$c_1^{m_i}, E^{O'(k, Enc_{q_1}(q_1,u_1),\ldots,Enc_{q_1}(q_d,u_d), q_1, .)}(U_n, pk_2, r'_E).$$

Consequently, $c_1^{m_i}, c_2^{m_i}$ is one-way in q because, by definition, the latter distribution is. Thus, $c_1^{m_i}, c_2^{m_i}$ can be regarding as auxiliary information about q and then by perfect one-wayness with auxiliary information, we have $c_0^{m_i}, c_1^{m_i}, c_2^{m_i}$ is computationally indistinguishable from $U_{|c_0^{m_i}|}, c_1^{m_i}, c_2^{m_i}$. However, as argued earlier, the latter distribu-

tion is indistinguishable from $U_{|c_0^{m_i}|}, c_1^{m_i}, E^{O'}(U_n, pk_2, .)$. We also know that $c_1^{m_i}$ does not reveal r^E because otherwise, it is easy to check whether $c_1^{m_i}$ is an encryption of m_i by recomputing $E(m_i, pk, r^E)$ and comparing it with $c_1^{m_i}$. Thus, we conclude that $U_{|c_0^{m_i}|}, c_1^{m_i}, E^{O'(.)}(U_n, pk_2, .)$ and consequently $c_0^{m_i}, c_1^{m_i}, c_2^{m_i}$ is one-way in r^E. Using perfect one-wayness again, we have $E'(m_i, pk', .) = H_k((m_i, r^E, c_0^{m_i}, c_2^{m_i}), .), c_0^{m_i}, c_1^{m_i}, c_2^{m_i}$ is computational indistinguishable from $U_{|H_k((m_i, r^E, c_0^{m_i}, c_2^{m_i}), .)|}, c_0^{m_i}, c_1^{m_i}, c_2^{m_i}$. The latter distribution is indistinguishable from $Extend(c_1^{m_i}, s) = U_{|H_k((m_i, r^E, c_0^{m_i}, c_2^{m_i}), .)|}, U_{|c_0^{m_i}|}, c_1^{m_i}$, $E^{O'}(U_n, pk_2, .)$. To finish the proof of Eq. 7.12, note that SIM does not help A in distinguishing $E'(m_i, pk')$ from $Extend(c, s)$ because SIM responds with $\perp$ except on ciphertexts for which A knows a corresponding plaintext.

Combining Eq. 7.11 and Eq. 7.12 implies that B has a nonnegligible advantage against semantic security of P. A contradiction. □

7.3.3 Realizing Unrealizable Schemes

[CGH98] contains a transformation that converts any encryption scheme to another one in the RO model with the following properties. The new scheme is secure if the original scheme is. However, any conventional instantiation fails to maintain security. It is interesting to note that if the original scheme is first-query hiding, both the noninteractive and interactive instantiation described in Section 7.3 can be modified to securely instantiate the new scheme. In other words, there are encryption schemes in the RO model for which there are secure (both IND-CPA and IND-CCA2) instantiations but no secure conventional instantiations.

We sketch here such an instantiation. Let P be any first-query hiding IND-CPA encryption scheme and P' be the scheme resulting from applying the transformation of [CGH98] to P. We apply a slightly different version of Construction 7.3.2 on $P' = (G', E', D')$ to get an IND-CCA2 scheme, P''. P'' differs from Construction 7.3.2 in that the first query of E, *not of* E' is used to encryption random oracles answers.

We emphasize that D'' does not reveal a plaintext that E'' or A does not know. In particular, D'' does not reveal the secret key as is the case with the conventional instantiation of P' (refer to the original paper for more detail).

7.4 OAEP

Even though OAEP [BR94] is not first-query hiding (see Definition 7.3.1), it has secure instantiations in very similar ways to Constructions 7.3.1 and 7.3.2.

Instead of encrypting Random Oracle answers using the first query, we encrypt the answers using the corresponding queries themselves. Then, we have the same results as Theorems 7.3.2 and 7.3.3.

We first formalize the original scheme and then give both the IND-CPA and IND-CCA2 instantiations.

Construction 7.4.1 (OAEP,[BR94]). *Let $\mathbb{M}$ be a trapdoor permutation (with key space PK_n and trapdoor space SK_n) and $P = (G, E, D)$ be the following encryption scheme in the RO model:*

- $G(1^n) = (pk, sk)$, *where* $pk \leftarrow PK_n$, $sk \leftarrow SK_n$.
- $E^{O_1,O_2}(m, pk) = M_{pk}(s, O_2(s) \oplus r)$, *where r is uniform and* $s = O_1(r) \oplus m$.
- $D^{O_1,O_2}(c, sk)$ *computes* $M_{sk}(c) = y_1, y_2$, *queries* $O_2(y_1)$, *computes* $r = y_2 \oplus O_2(y_1)$, *queries* $O_1(r)$, *and recover* $m = y_1 \oplus O_1(r)$.

Construction 7.4.2 (IND-CPA instantiation). *Let $\mathbb{M}$ be a trapdoor permutation (with key space PK_n and trapdoor space SK_n), $\mathbb{H}$ be a verifiable family ensemble, and $P' = (G', E', D')$ be the following encryption scheme in the standard model:*

- $G'(1^n) = (pk, sk, k)$, *where* $pk \leftarrow PK_n$, $sk \leftarrow SK_n$, *and* $k \leftarrow K_n$.
- $E'(m, pk, k) = Enc_r(r, u_1), Enc_s(s, u_2), M_{pk}(s, u_2 \oplus r)$, *where r is uniform,* $s = u_1 \oplus m$, $Enc_i(j, k) = H_k(i, j, r_0), H_k(i, j, k_1, r_1), ..., H_k(i, j, k_n, r_n)$, *and $r_0, ..., r_n$ are uniform.*
- $D'(c = (c_0, c_1), sk, k)$ *computes* $M_{sk}(c_1) = y_1, y_2$, *recovers* u_2 *from* y_1 *and* c_0 *(in particular* $Enc_s(s, u_2)$*), computes* $r = y_2 \oplus u_2$, *recovers* u_1 *from* r *and* c_0 *(in particular* $Enc_r(r, u_1)$*), and outputs* $m = y_1 \oplus u_1$.

Using the same arguments as in Theorem 7.3.2, we show that Construction 7.4.2 is IND-CPA, provided that the trapdoor permutation used in OAEP is partially one-way. Recall that a function is partially one-way if it is one-way in the first half of the input

and one-way in the second half as well. The proof of the following theorem is almost the same as that of Theorem 7.3.2 and is omitted here.

Theorem 7.4.1. *Let $\mathbb{M}$ be a partially one-way trapdoor permutation and $\mathbb{H}$ be any family ensemble that satisfies strong pseudorandomness with auxiliary information (as in Definition 4.3.5), and collision resistance (as in Definition 2.5.2). Then, Construction 7.4.2 is IND-CPA (as in Definition 2.7.1).*

We strengthen Construction 7.4.2 in a similar way to Construction 7.3.2 to get a CCA2-secure instantiation against a constant number of decryption queries. Formally,

Construction 7.4.3 (IND-CCA2 instantiation). *Let $\mathbb{M}$ be a trapdoor permutation (with key space PK_n and trapdoor space SK_n), $\mathbb{H}$ be a verifiable family ensemble, and $P' = (G', E', D')$ be the following encryption scheme in the standard model:*

- $G'(1^n) = (pk_1, pk_2, sk_1, sk_2, k)$, *where* $pk_1, pk_2 \leftarrow PK_n$, $sk_1, sk_2 \leftarrow SK_n$, *and* $k \leftarrow K_n$.
- $E'(m, pk_1, pk_2, k) = H_k((m, r, c_0, c_2), c_0, c_1, c_2$

$$c_0 \triangleq Enc_r(r, u_1), Enc_r(r', u_1'), Enc_r(s', u_2'), Enc_s(s, u_2)$$

$$c_1 \triangleq M_{pk_1}(s, u_2 \oplus r)$$

$$c_2 \triangleq M_{pk_2}(s', u_2' \oplus r'),$$

  where $r, r', u_1, u_2, u_1', u_2'$ are uniform, $s = u_1 \oplus m$ and $s' = u_1' \oplus r$.

- $D'(c = (y, c_0, c_1, c_2), sk_1, sk_2, k)$
  - *compute $M_{sk_1}(c_1) = y_1, y_2$ and $M_{sk_2}(c_2) = y_1', y_2'$*
  - *recover u_2 from y_1 and c_0 (in particular $Enc_s(s, u_2)$)*
  - *compute $r = y_2 \oplus u_2$,*
  - *recover u_1 from r and c_0*
  - *compute $m = y_1 \oplus u_1$*
  - *recover u_2' from r, y_1', c_0*
  - *compute $r' = y_2' \oplus u_2'$*

- *recovers* u'_1 *from* r, r', c_0
- *compute* $r'' = y'_1 \oplus u'_1$
- *if* $r'' \neq r$ *or* $V_{\mathbb{H}}((m, r, c_0, c_2), y) \neq 1$ *or* c_0 *is not a valid encryption of* u_1, u_2, u'_1, u'_2*, return* $\perp$
- *return* m

The following theorem is a specialization of Theorem 7.3.3 to OAEP.

Theorem 7.4.2. *Let* $\mathbb{M}$ *be a partially one-way trapdoor permutation, and* $\mathbb{H}$ *be any family ensemble that satisfies strong pseudorandomness with auxiliary information (as in Definition 4.3.5), extraction with auxiliary input (as in Definition 3.2.5), and collision resistance (as in Definition 2.5.2). Then, Construction 7.4.3 is IND-CCA2 (as in Definition 2.7.2), against a constant number of decryption queries.*

Chapter 8

Digital Lockers: Obfuscating Multibit Point Functions

Summary: We study obfuscation of point functions with multibit output and other related functions. A point function with multibit output returns a fixed string on a single input point and zero everywhere else. Obfuscation of such functions has a useful application as a strong form of symmetric encryption where security holds without any assumption on the distribution of the secret key. We provide a construction that obfuscates these functions. The construction is generic in the sense that it can use any perfectly one-way (POW) function or obfuscator for point functions.

Analyzing this construction reveals gaps in the definition of obfuscation, specifically, that it does not guarantee security even under self-composition, a property needed in our analysis. Thus, we use obfuscation secure under composition. In particular, we show that composable obfuscation of multibit point functions exists if and only if composable obfuscation of point functions exists. Moreover, we show that this construction is secure based on statistically indistinguishable POW functions. However, if we relax the assumption to computational indistinguishability, then the construction satisfies a weaker notion of obfuscation. Finally, the same technique can be used to obfuscate set-membership predicates and functions, for polynomial-size sets.

This chapter is based on the paper [CD08b], which is a joint work with Ran Canetti.

8.1 Introduction

Obfuscation is one of the most intriguing problems in cryptography. Informally, an obfuscator is a compiler that converts a program into another one, called the obfuscated program or code, that has a similar functionality but satisfies certain secrecy requirements. Informally, the secrecy requirement stipulates that whatever "useful" information the obfuscated code reveals is learnable from the program's input/output behavior. In other words, an obfuscated program should not reveal anything useful beyond executing it. This requirement is formalized by Barak *et al.* [BGI+01] through a simulation-based definition called the virtual-blackbox property. The virtual-blackbox property says that every adversary has a corresponding simulator that emulates the output of the adversary given oracle (i.e., blackbox) access to the same functionality being obfuscated.

In the same work, Barak *et al.* provide impossibility results regarding general obfuscation, even when the output of the adversary is restricted to predicates. In other words, it is shown that there are certain functionalities and corresponding predicates where these predicates are learnable from any program implementing the functionalities but not so given blackbox access to them. In light of this general negative result, we are forced to study obfuscation of restricted classes of functions if we wish to adopt the definition of [BGI+01]. Here, we follow this line of work. In particular, we build on previous work on point function obfuscation [Can97, CMR98, Wee05, LPS04] towards obfuscating slightly more complex functions, namely point functions with multibit output. Moreover, we show that obfuscation of point functions are not necessarily secure even under self-composition, a property needed in our analysis. We next go into a more detailed exposition of our work.

8.1.1 Our Work

Obfuscation of point functions with multibit output. A point function returns 1 on a single input and 0 everywhere else. Formally, $F_x(y) = 1$ if $y = x$ and 0 otherwise. A point function with multibit output generalizes point functions in that it outputs, on a single input, a long string instead of 1. Formally, $F_{x,y}(z) = y$ if $z = x$, and 0 otherwise.

The connection to symmetric encryption. Obfuscation of such functions has a useful application as what we call a **digital locker**. A digital locker is a strong form of

symmetric encryption where privacy holds without an assumption on the distribution of the key. Privacy without requiring anything about the distribution of the key essentially means that nothing can be learned about the plaintext unless the key is recovered in full. That is, the complexity of learning anything about the plaintext corresponds to that of finding the key. We stress that this notion is not ruled out by the impossibility results of [MP90, DS02, BD07] because we allow the encryption scheme to be probabilistic and thus, has access to a perfectly random source.

Real life applications of such a notion include password-based encryption where the human-generated password is far from uniform. For instance, Firefox has a password manager that acts as a digital locker [FPM]. The password manager locks website credentials using a master password chosen by the user. Then, the user has to provide this password in order to unlock the content.

We formalize this privacy notion using the simulation paradigm in a way similar to obfuscation. Specifically, the behavior of the adversary on a ciphertext is simulated given blackbox access to the multibit point function, $F_{key,plaintext}$. Thus, obfuscation of point functions with multibit output can be used to realize digital lockers as follows: to encrypt a message m using a key k, simply output the obfuscation of $F_{k,m}$.

A closer look at our definition of digital lockers reveals the following weakness. Even though privacy is captured when the secret key is uniform or taken from a well-spread distribution (i.e., the min-entropy is superlogarithmic), the definition does not really capture privacy when the distribution is not well-spread, e.g., when the support is of polynomial size. This is so because it does not relate the number of queries of the simulator to the running time of the adversary. Consequently, an anomaly arises. A scheme, deemed secure by this definition, may reveal the plaintext when the key is taken from a polynomial-sized set. Note that this weakness is not restricted to this application. Rather, it applies to obfuscation in general: an obfuscation may be totally insecure on a polynomial number of functions. We explore one way to address this weakness. Further work on this issue is left for further research.

The construction. Even though obfuscation of point functions with multibit output is known in the Random Oracle Model [LPS04], it is not known in the standard model except when the function is drawn from a uniform distribution (specifically, when x in $F_{x,y}$ is uniform) [FKSW05] or when the output length of the function is short (specif-

ically, when $|y| = O(log|x|))$ [Wee05]. Here, we provide a transformation from point function obfuscators to obfuscators of point functions with multibit output. The idea is simple. The obfuscation of multibit point functions consists of some number of copies of obfuscated point functions. These copies have the property that the first and the ith copy correspond to an obfuscation of the same point function if and only if the ith bit in the multibit output is 1. In more detail, let $F_{a,b}$ be the multibit point function to be obfuscated, $t = |b|$, and $O(F_a, r)$ be the obfuscation of the point function, F_a, using randomness r. Then, the obfuscation of $F_{a,b}$ consists of $O(F_a, r_0), O(x_1, r_1), ..., O(x_t, r_t)$, where x_i is F_a if $b_i = 1$ and x_i is a uniformly chosen point function otherwise. To recover b from the correct a and this obfuscation, first verify that $O(F_a, r_0)(a) = 1$, then $b = O(x_1, r_1)(a), ..., O(x_t, r_t)(a)$.

On composing obfuscation. The construction described above is very simple and modular, and one expects that its proof be likewise. However, it turns out that this is not the case. To prove the security of the above transformation, we face an issue. Observe that our construction is composed of a concatenation of $t + 1$ obfuscated point functions. Thus, in order for our construction to be secure, the original obfuscation *has* to remain secure under composition. However, we show that the current definition of obfuscation does not guarantee composition. This is also the case even for composing multiple obfuscated copies of the *same* function. Interestingly, the statement still holds even if we consider obfuscation secure in the presence of auxiliary information. We emphasize that this is a fundamental point about the definition of obfuscation that is of independent interest.

In more detail, we show that there exists an obfuscation of point functions that reveals the input when it is self-composed. Specifically, we show an obfuscator, O, such that for any x, it is possible to recover x from $O(F_x, r_1), ..., O(F_x, r_{nlog(n)})$, where $n = |x|$. Moreover, similar results holds for POW functions and POW functions with auxiliary information [Can97, CMR98].

In light of these negative results, we analyze the above construction using, as the underlying primitive, three different forms of composable obfuscation of point functions. First, if the underlying primitive is a composable obfuscation of point functions (as in the simply-composable obfuscation of [LPS04]), then this construction is a composable obfuscation of multibit point functions. This is actually a characterization: composable

obfuscation of point functions exists if and only if that of multibit point functions exists. Second, we show that our construction is an obfuscation of multibit point functions if the underlying primitive is a statistically indistinguishable POW function.[1] Third, if the primitive is a computationally indistinguishable POW function, then the construction is an obfuscation provided that y in $F_{x,y}$, is independent of x (see Eq. 8.3).

Finally, we show how to generalize this construction to obfuscate set-membership predicates and functions for polynomial-sized sets. A set-membership predicate outputs 1 if the input belongs to the set and 0 otherwise, while a set-membership function outputs a string, y_i, if the input matches a set member, x_i, and 0 otherwise.

8.1.2 Related Work

Obfuscating Point Functions in the Random Oracle Model. Lynn *et al.* [LPS04], inspired by the password-hiding scheme in Unix that stores a hash of the password instead of the password itself, propose a similar obfuscation of point functions in the random oracle model. In this model, an obfuscator, O, has oracle access to a truly random function, R. In order to construct an obfuscation of a point function, F_x, O queries R on x to get $z = R(x)$ and then stores z in the obfuscated code, $O(F_x)$. $O(F_x)$ also contains preprocessing code which on input y returns 1 if and only if $R(y) = z$.

It is easy to see that $O(F_x)$ and F_x have approximate functionality (they have the same functionality almost always). Intuitively, $O(F_x)$ is an obfuscation of F_x because R's answers on queries are completely independent and random. So, storing $R(x)$ does not reveal any information about x, but it allows verification of a guess, which is also achievable via oracle access to F_x.

Also, Lynn *et al.* [LPS04] generalize this construction to obfuscate multibit point functions and set-membership predicates and functions in the random oracle model. To obfuscate a multibit point function, $F_{x,y}$, choose a random r, and output $r, R_1(x,r), R_2(x,r) \oplus y$, where R_1 and R_2 denote the first and second half of the bits of $R(.)$. This construction is secure under composition (as in Definition 8.2.1 or the simply-composable definition of [LPS04]). In Section 8.2.3, we instantiate this scheme. The resulting construction is more efficient than our first one but uses a stronger assumption.

[1] To be accurate, the second construction satisfies approximate functionality only computationally, i.e., it is hard to efficiently find an input point on which the obfuscated function differs from the original one.

Obfuscating Point Functions in the standard model. Perfectly one-way (POW) functions [Can97] can be used to obfuscate a point function F_x by replacing the random oracle in [LPS04] with a POW function, H. Here, instead of storing $R(x)$, we store $H(x)$ in the obfuscated code and use the verifier for H to determine if $H(x)$ is a valid hash of the input.

Canetti [Can97] constructs a POW function based on a strong version of the Diffie-Hellman assumption. In particular, it assumes that the Diffie-Hellman assumption holds not only against uniform distributions but also with respect to any well-spread distribution (see Assumption 3.3.2). Moreover, Wee [Wee05] shows how to obfuscate point functions and point functions with logarithmic output based on a strong one-way permutation assumption. Specifically, the assumption is that any polynomial-time machine can invert the permutation on at most a polynomial number of points. The two constructions mentioned so far (and our construction as well) use a weaker notion of obfuscation than the one in [BGI+01]. Specifically, the simulator in [Can97, Wee05] depends on the simulation-error gap between the adversary and the simulator (see Definition 2.6.1 for more detail).

Canetti *et al.* [CMR98] provide two constructions of POW functions based on standard computational assumptions (in particular, based on either claw-free permutations or one-way permutations). The simulator for these constructions does not depend on the gap. However, the input distribution is assumed to have high min-entropy (n^ϵ). Moreover, Futoransky *et al.* [FKSW05] show how to obfuscate point functions and point functions with multibit output based on standard assumption. However, the input distribution is assumed to be uniform. Finally, Hofheinz *et al.* [HMLS07] obfuscate point functions *deterministically*. However, the secrecy requirement does not guarantee no information leakage, rather that it is hard to recover the input in its entirety. This obfuscation is self-composable because the obfuscator is deterministic. However, it is not composable according to our notion. In particular, different obfuscated point functions can not be securely composed.

8.1.3 Organization

We present our construction and analyze it in Section 8.2. (We also present a more efficient construction under a stronger assumption in Section 8.2.3.) In Section 8.3, we study the issue of composable obfuscation. Finally, we discuss the connection to encryption schemes in Section 8.4.

8.2 Obfuscating Point Functions with Multibit Output

We show how to obfuscate point functions with multibit output as well as set-membership predicates and functions for polynomial-sized sets. Because the constructions and proofs for obfuscating set-membership predicates and functions are similar to that for multibit point function, we focus on the latter. We comment on the former in Section 8.2.2. Finally, we present a more efficient obfuscation of multibit point functions using a stronger assumption in Section 8.2.3.

We use obfuscated point functions as building blocks in obfuscating point functions with multibit output. The idea is simple. To obfuscate $F_{x,y}$, we encode y bit-by-bit using an obfuscator for F_x. Specifically, if the ith bit of y is 1, it is encoded as an obfuscation of F_x, otherwise, it is encoded as an obfuscation of an independent and uniform point function. In more detail, let H be a randomized obfuscator for point functions. Then the obfuscation contains $H(F_x, r), H(F_{x_1}, r_1), ..., H(F_{x_t}, r_t)$, where $t = |y|$ and $x_i = x$ if the ith bit of y is 1, otherwise, x_i is uniform. The first obfuscated point functions always corresponds to x, and is used to check whether the input is actually x. Now, y can be recovered given $z = x$. First, check that $H(F_x, r)(z) = 1$. If so, for every i, $y_i = H(F_{x_i}, r_i)(z)$.

Formally, we present an obfuscator, O, for the class of multibit output point functions, $\mathbb{F}$. O, on input $F_{x,y}$, where y has length t, selects $r_1, ..., r_{t+1}$ from R_n, the randomness domain of the point function obfuscator, H. It then computes $H(F_x, r_1)$. It also computes $H(F_x, r_{i+1})$ if $y_i = 1$ and $H(z_{i+1}, r_{i+1})$ otherwise, where z_{i+1} is uniform. Let $u_x = u_1, ..., u_{t+1}$ be the sequence of obfuscated functions just computed. Then O

outputs the following obfuscation, $O(F_{x,y})$, with u_x stored in it.

input: a

```
1  if u_1(a) = 0 then
2      return 0;
3  else
4      for i ← 2 to t + 1 do
5          if u_i(a) = 1 then
6              y_{i-1} ← 1;
7          else
8              y_{i-1} ← 0;
9          return y = y_1, ..., y_t;
10     end
```

Algorithm 8.2.1: $O(F_{x,y})$

8.2.1 Analysis

This construction is simple and modular. It is possible to replace H by any relative of point function obfuscation such as POW functions and analyze the security of the construction based on the security of the underlying primitive. We would like to prove that our construction is secure based on the simple assumption that the underlying primitive is an obfuscation of point functions. However, as we show in Section 8.3, this is not possible. This is so because the definition of obfuscation does not guarantee even self-composition. Thus, there exist point function obfuscators and POW functions for which this construction is provably insecure.

We investigate the secrecy of this construction based on three underlying primitives with different composition properties. In the first case, we consider the notion of composable obfuscation (as in Definition 8.2.1, also known as simply-composable obfuscation in [LPS04]). We give a characterization that shows that composable point function obfuscation exists if and only if composable multibit point function obfuscation exists. In the second case, we show that if H is a statistically indistinguishable POW function, then our construction is secure. Finally, if H is a computationally indistinguishable POW then this construction satisfies a weaker form of obfuscation where y, in $F_{x,y}$, is independent of x.

8.2.1.1 Analysis based on composable obfuscation

In this work, composable obfuscation refers to the fact that concatenating any sequence of obfuscated functions, where the functions are taken from the same class, constitutes an obfuscation for that sequence of functions. This form of composition, also known as simply-composable obfuscation in [LPS04], should not be confused with self-composition, which means that concatenating a sequence of independent obfuscation of the *same* function does not compromise secrecy. Formally,

Definition 8.2.1 (t-Composable Obfuscation, [LPS04]). *Let $\mathbb{F}$ be any family of functions. A PPT, O, is called a t-composable obfuscator for $\mathbb{F}$, if:*

1. *Approximate functionality and polynomial slowdown are as in Definition 2.6.1.*
2. ***Virtual Black-box property*** *For any nonuniform PPT, A, and any polynomial, p, there is a nonuniform PPT, S, such that for any functions $F_1, ..., F_{t(n)} \in \mathbb{F}$ (n is a security parameters, e.g., $n = |F_1| = ... = |F_{t(n)}|$) and sufficiently large n:*

$$|Pr[b \leftarrow A(O(F_1), ...O(F_{t(n)}) : b = 1] - Pr[b \leftarrow S^{F_1,...,F_{t(n)}}(1^n) : b = 1]| \leq \frac{1}{p(n)}$$

If O is a t-composable obfuscator for $\mathbb{F}$ for any polynomial t, then it is called a composable obfuscator.

If H is a $(t+1)$-composable obfuscator for some t, then our construction can be shown to be an obfuscation of multibit point function with output length t. Approximate functionality and polynomial slowdown follow from the corresponding properties on H. By the virtual black-box property on H, the output of $A(O(F_{x,y}) = O(F_x), O(F_{x_1}), ..., O(F_{x_{t(n)}}))$ can be simulated by $S^{F_x,F_{x_1},...,F_{x_{t(n)}}}(1^n)$, where $x_i = F_x$ if $y_i = 1$ and x_i is uniform otherwise. Moreover, oracle access to $F_x, F_{x_1}, ..., F_{x_{t(n)}}$ can be simulated with oracle access to $F_{x,y}$: if S queries any of its oracle on a point z such that $F_{x,y}(z) = 0$, then answer 0 (this may incur a negligible simulation error only), otherwise, $z = x$ so y can be fully recovered. Thus, this construction satisfies the virtual black-box property.

Observe that our construction is a *composable* obfuscation of multibit point functions with the appropriate parameters. Specifically, if the output length of the multibit point function is restricted to at most t, then this construction is a t'-composable obfuscation

if H is $(t+1)t'$-composable. In addition, it is easy to see that the existence of a t-composable obfuscation of multibit point functions implies a t-composable obfuscation of point functions. Formally, we have the following characterization with a proof that follows the above discussion.

Theorem 8.2.1. *Composable obfuscators of point functions with multibit output exist if and only if composable obfuscators of point functions exist.*

Specifically, if a point function obfuscator, H, is $(t+1)t'$-composable (as in Definition 8.2.1) then the above construction is a t'-composable obfuscation of multibit point functions with output length t. On the other hand, a t-composable obfuscation of multibit point functions implies a t-composable obfuscation of point functions.

8.2.1.2 Analysis based on statistical indistinguishability

Suppose $\mathbb{G}$ is a statistically indistinguishable POW family ensemble (as in Definition 2.5.3). We can replace H by $\mathbb{G}$ in the above construction. Specifically, the obfuscator, O, samples a key, k, for $\mathbb{G}$ and replaces $H(x,.)(a)$ with $V(a, G_k(x,.))$, where V is the verification algorithm for $\mathbb{G}$. This results in an obfuscation of point function with multibit output except with *computational approximate functionality* [Wee05], i.e, no adversary can efficiently find a point on which the original function differs from the obfuscated one. This relaxation to approximate functionality is necessary when using statistical POW functions because they can not be statistically collision resistant. Formally,

Definition 8.2.2 (Computational functionality). *Let $\mathbb{F}$ be any family of functions. A PPT, O, is called an **obfuscator** of $\mathbb{F}$, with computational functionality if for any $F \in \mathbb{F}$ and any nonuniform PPT, A: $Pr[x \leftarrow A(O(F)) : O(F)(x) \neq F(x)] \leq \mu(n)$.*

On the other hand, we argue that the result satisfies the virtual-blackbox property. Informally, from the fact that $\mathbb{G}$ is a statistical POW function we can conclude that an obfuscation of $F_{x,y}$, where x is taken from a well-spread distribution and y is arbitrary, is statistically close to a sequence of images of random inputs. It follows that for all but polynomially many x, an obfuscation of $F_{x,y}$ is indistinguishable from random images. Consequently, we get a simulator that runs the adversary on random images unless x is taken from this polynomial set, in which case the simulator can recover y and run the adversary on an obfuscation of $F_{x,y}$. Formally,

Theorem 8.2.2. *Let $\mathbb{G}$ be a statistically $(t+1)$-indistinguishable POW function (as in Definition 2.5.3) with public verification and collision resistance (as in Definitions 2.5.1 and 2.5.2). Then, the above construction is an obfuscation of point functions with multibit output length t, where approximate functionality is only computational (as in Definitions 2.6.1 and 8.2.2).*

Proof. **Polynomial slowdown.** This follows immediately from the fact that $\mathbb{G}$ has a polynomial output length.

Computational approximate functionality. Suppose for the purpose of contradiction there is a function, $F_{x,y}$ and a nonuniform PPT, A that violates computational functionality, i.e., $Pr[x' \leftarrow A(O(F_{x,y})) : O(F_{x,y})(x') \neq F_{x,y}(x')]$ is nonnegligible. Let B be the following adversary that defeats collision resistance of $\mathbb{G}$. Then, B has $F_{x,y}$ as auxiliary information and receives G_k. It simulates O, using G_k (i.e., it skips the step where O samples G_k), on $F_{x,y}$ to get $O(F_{x,y})$ which contains $w = G_k(x,r)$ for some key k and random r. It then runs A on $O(F_{x,y})$ to compute x' and outputs (x,x',w). By construction $V_{\mathbb{G}}(x,w) = 1$. Also, since $O(F_{x,y})(x') \neq F_{x,y}(x')$ then this means that $V_{\mathbb{G}}(x',w) = 1$. Note that the input to A when simulated by B is equivalent to that in the definition of approximate functionality. Consequently, B outputs a collision with the same probability that A finds an input on which $F_{x,y}$ and $O(F_{x,y})$ differ. This contradicts collision resistance.

Virtual black-box property. Recall, the definition of statistical indistinguishability says that for any well-spread distribution, $\mathbb{X}$:

$$\Delta(G_k(X_n,R_n^1),...,G_k(X_n,R_n^{(t+1)(n)}),G_k(U_n^1,R_n^1),...,G_k(U_n^{t(n)},R_n^{(t+1)(n)}) \leq \mu(n),$$

where each distribution R_n^i (respectively, U_n^i) is the same as R_n (respectively, U_n).

Using the fact that for any function, λ, $\Delta(\lambda(X),\lambda(Y)) \leq \Delta(X,Y)$, we have for any distribution,$\mathbb{X}\mathbb{Y}$ on (x,y), where the corresponding distribution on x is well-spread:

$$\Delta(O(F_{X_n,Y_n}),G_k(U_n^1,R_n^1),...,G_k(U_n^{t(n)},R_n^{(t+1)(n)}) \leq \mu(n). \tag{8.1}$$

(We assume without loss of generality that $O(F_{x,y})$ consists only of the $t+1$ $\mathbb{G}$-images.)

Using the same technique from the proof of Theorem 4 in [Can97], it can be shown that $O(F_{x,y})$ is indistinguishable from $\mathbb{G}$-images of uniform strings on all but a polynomial number of x. That is, for any nonuniform PPT, A, and any polynomial, p, there exists a family of polynomial-size sets, $\{L_n\}$, such that for sufficiently large n, $x \notin L_n$, and any y:

$$|Pr[b \leftarrow A(O(F_{x,y})) : b = 1] -$$

$$Pr[u_1, ..., u_{t+1} \leftarrow U_n, ..., U_n,$$

$$r_1, ..., r_{t+1} \leftarrow R_n, ..., R_n,\ b \leftarrow A(G_k(u_1, r_1), ..., G_k(u_{t+1}, r_{t+1})) : b = 1]| \leq \frac{1}{p(n)}. \quad (8.2)$$

Intuitively, this is true because otherwise there is a super-polynomial number of values for x (with a corresponding value for y), on which A can distinguish $O(F_{x,y})$ from images of random strings. By defining a well-spread distribution, e.g., a uniform distribution, on this superpolynomial number of values for x, A violates Eq. 8.1. For the complete proof of Eq. 8.2, we refer the reader to [Can97].

Now, for any nonuniform PPT, A, and a polynomial, p, we construct a nonuniform PPT, S that simulates A. S receives the polynomial-size set, L_n, as an advice string. It checks if the oracle, $F_{x,y}$, responds with the nonzero value, y, to any element in the set, L_n. If so, then S can compute $O(F_{x,y})$ and simulate A on it. Otherwise, x is not in L_n, so S runs A on images of random inputs. By Eq. 8.2, this is close to a true simulation. □

8.2.1.3 Analysis based on computational indistinguishability

We would like to weaken the assumption in Theorem 8.2.2 to computational indistinguishability. However, it is not clear how to use computational indistinguishability, i.e., that $G_k(x, r_1), ..., G_k(x, r_{t+1})$ is computationally indistinguishable from images of uniform-distributed inputs, to conclude that $O(F_{x,y})$ is indistinguishable from images of random inputs. It seems that the problem lies in the potential dependence of y on x, e.g., y may be equal to x. This is not a problem in the statistical case, because we can use the fact that statistical difference does not increase by applying the same function on both distributions. In the computational setting, if we use traditional blackbox reductions, we need to construct $O(F_{x,y})$ from images of x and then run A on it. However, it is not clear

how to do so if $y = x$. On the other hand, suppose y is independent of x. Then, for some y, it is possible to compute $O(F_{x,y})$ given images of x, $G_k(x, r_1), ..., G_k(x, r_{t+1})$, by replacing $G_k(x, r_i)$ with an image of a random string if the ith bit of y is 0. Thus, we know that computational indistinguishability gives us a weaker notion of obfuscation where the simulator depends on the distribution on y. Whether computational indistinguishability gives us the standard virtual-blackbox property remains unknown. Nevertheless, this weak obfuscation can be used as a digital locker as described in the introduction. The caveat is that the message being encrypted should be independent of the encryption key. This is the case if, for instance, the message is chosen without knowledge of the key.

Formally, the virtual black-box property becomes: for any nonuniform PPT A, any polynomial p, and any (efficiently samplable) distribution $\mathbb{Y}$, there exists a nonuniform PPT S such that for any x and sufficiently large n:

$$|Pr[y \leftarrow Y_n,\ b \leftarrow A(O(F_{x,y})) : b = 1] - Pr[y \leftarrow Y_n,\ b \leftarrow S^{F_{x,y}}(1^{|F_{x,y}|}) : b = 1]|$$

$$\leq \frac{1}{p(n)}. \tag{8.3}$$

Also, we remark that this construction has either approximate or computational approximate functionality depending on whether the POW function satisfies statistical or computational collision resistance. Formally, we have the following theorem whose proof follows that of Theorem 8.2.2 and the above discussion.

Theorem 8.2.3. *If $\mathbb{G}$ is a computationally $(t+1)$-indistinguishable POW function (as in Definition 2.5.5, against nonuniform adversaries) with public verification and collision resistance (as in Definition 2.5.2), then the above construction is an obfuscation of point function with output length t (as in Definition 2.6.1, where the virtual-blackbox property is as in Eq. 8.3).*

Proof. **Polynomial slowdown.** This follows immediately from the fact that $\mathbb{G}$ has a polynomial output length.

Computational functionality. Same proof as Theorem 8.2.2. Moreover, (statistical) approximate functionality can be proven in the same way by removing the polynomial-time restriction on adversaries.

Virtual Blackbox property. Let $\mathbb{G}$ be a computational $t+1$-indistinguishable POW

function. For simplicity, we remove any preprocessing code from $O(F_{x,y})$ and view it as $G_k(x,r), G_k(x_1,r_1), ..., G_k(x_t,r_t)$, where $x_i = x$ if $y_i = 1$, otherwise x_i is uniform.

First, we claim that $O(F_{x,y})$ is computationally indistinguishable from images of uniform strings, where x is taken from any well-spread distribution and y is taken from Y_n. Then we use the proof idea of Theorem 4 in [Can97], to show that $O(F_{x,y})$ is computationally indistinguishable from images of uniform strings on all but a polynomial number of x. Finally, we hardwire this polynomial-sized set of inputs into a simulator.

In more detail, we have for any nonuniform PPT, A, any well-spread distribution $\mathbb{X}$, and any $k \in K_n$:

$$|Pr[x \leftarrow X_n,\ y \leftarrow Y_n,\ b \leftarrow A(O(F_{x,y})) : b = 1] -$$

$$Pr[x \leftarrow X_n\ u_1, ..., u_{t+1} \leftarrow U_n, ..., U_n,\ r_1, .., r_{t+1} \leftarrow R_n, ..., R_n,$$

$$b \leftarrow A(G_k(u_1,r), ..., G_k(u_{t+1}, r_{t+1})) : b = 1]| \leq \mu(n). \tag{8.4}$$

In the above inequality, we assume that O uses G_k. Suppose, for the purpose of contradiction that Eq. 8.4 is not true. Let A be an adversary that defeats it. Consider another nonuniform PPT, B, that receives $t+1$ images under $\mathbb{G}$, $w_1, ..., w_{t+1}$. B samples y from Y_n and simulates A on $w_1, w_2', ..., w_{t+1}'$, where $w_i' = w_i$ if $y_i = 1$, otherwise $w_i' = G_k(u_i, r_i)$, where u_i and r_i are sampled uniformly by B. Observe that if $w_1, ..., w_{t+1}$ are images of the same input, x, then B simulates A on $O(F_{x,y})$. However, if $w_1, ..., w_{t+1}$ are images of uniform and independent inputs, then B simulates A on images of uniform and independent inputs as well. Therefore, we have by Eq. 8.4:

$$|Pr[x \leftarrow X_n,\ r_1, ..., r_{t+1} \leftarrow R_n, ..., R_n,\ b \leftarrow B(G_k(x,r_1), ..., G_k(x,r_{t+1})) : b = 1] -$$

$$Pr[u_1, ..., u_{t+1} \leftarrow U_n, ..., U_n,\ r_1, ..., r_{t+1} \leftarrow R_n, ..., R_n,$$

$$b \leftarrow B(G_k(u_1,r_1), ..., G_k(u_{t+1}, r_{t+1})) : b = 1]|$$

is nonnegligible. This contradicts computational indistinguishability on $\mathbb{G}$.

By Eq. 8.4 and the proof of Theorem 4 in [Can97], we have for any nonuniform PPT, A, and any polynomial p, there exists a family of polynomial-size sets, $\{L_n\}$, such that

for sufficiently large n, and $x \notin L_n$ and any y:

$$|Pr[b \leftarrow A(O(F_{x,y})) : b = 1] -$$

$$Pr[u_1, ..., u_{t+1} \leftarrow U_n, ..., U_n,$$

$$r_1, ..., r_{t+1} \leftarrow R_n, ..., R_n,\ b \leftarrow A(G_k(u_1, r_1), ..., G_k(u_{t+1}, r_{t+1})) : b = 1]| \leq \frac{1}{p(n)}.$$

The remaining part of the proof is exactly the same as the last part of the proof of Theorem 8.2.2. □

8.2.2 Obfuscating Set-membership Predicates and Functions

To obfuscate a set-membership predicate, simply obfuscate the point functions on every element in the set (this is feasible because the set has a polynomial size), and then store all the obfuscated functions in a randomly permuted order. To determine whether a particular input is in the set, we only need to check whether any of the obfuscated functions outputs 1 on this input. It can be shown, in a direct way, that this construction is an obfuscation of set-membership predicate based on composable obfuscation of point functions. In fact, composable obfuscation of point functions is also an obfuscation of set-membership predicates.

Moreover, to obfuscate a set-membership function, $F_{(x_1,y_1),...,(x_t,y_t)}$, we only need to run the obfuscator for the multibit output point function on each F_{x_i,y_i}, and then store these obfuscated functions in a randomly permuted order. Theorem 8.2.1 implies that this construction is secure if the underlying obfuscation is composable.

8.2.3 A More Efficient Obfuscation of Multibit Point Functions

We note that the obfuscation of multibit point function in the RO model [LPS04] can be instantiated by using a stronger assumption on the underlying primitive. The end result is a more efficient construction than the one described previously. Specifically, let $\mathbb{G}$ be a POW function with public randomness. To obfuscate $F_{x,y}$, select r_1 and r_2 uniformly from the randomness domain of $\mathbb{G}$ and output $G_k(x, r_1), r_2, z$, where $G_k(x, r_2) = (r_2, v)$ and $z = y \oplus v$.[2] To recover y from (a, b, c) and x', first check that $V(x', a) = 1$, if so, then

[2]Without loss of generality, we assume that y and v have the same length. Otherwise, the input should be longer, say $x0^t$.

return $y = c \oplus v$, where $G_k(x', b) = (b, v)$. Even though this construction is more efficient than the first one, it suffers from two problems. First, in order to completely hide y, it is not sufficient that $\mathbb{G}$ be indistinguishable as in Definition 2.5.5 rather its output has to be *indistinguishable from uniform*. If, for example, the first bit of the image is always 0, then the first bit of y is revealed. Second, if y is allowed to depend on x, we need to assume that $\mathbb{G}$ is *statistically* indistinguishable from *uniform*. Contrast this assumption with the one used in Theorem 8.2.2, where $\mathbb{G}$ is statistically indistinguishable from *images of uniform strings*.

Theorem 8.2.4. *Let $\mathbb{G}$ be a statistically 2-pseudorandom POW function (as in Definition 2.5.4) with public randomness and collision resistance (as in Definition 2.5.2). Then, the above construction is an obfuscation of point functions with output length $l(n) - |r|$ ($|x| = n$, $l(n) = |G_k(x)|$), where approximate functionality is only computational (as in Definitions 2.6.1 and 8.2.2).*

Proof. **Polynomial slowdown.** This follows immediately from the fact that $\mathbb{G}$ has a polynomial output length.

Computational approximate functionality. Same as that of Theorem 8.2.2.

Virtual black-box property. From the definition of statistical pseudorandomness, we have for any well-spread distribution, $\mathbb{X}$:

$$\Delta(G_k(X_n, R_n^1), G_k(X_n, R_n^2), U_{2l(n)}) \leq \mu(n),$$

where each distribution R_n^i is the same as R_n.

Using the fact that for any function, λ, $\Delta(\lambda(X), \lambda(Y)) \leq \Delta(X, Y)$, we have for any distribution,$\mathbb{XY}$ on (x, y), where the corresponding distribution on x is well-spread:

$$\Delta(O(F_{X_n, Y_n}), U_{2l(n)}) \leq \mu(n). \tag{8.5}$$

From here on, the proof is the same as that of Theorem 8.2.2. □

8.3 On Composable Obfuscation of Point Functions

In Section 8.2, we provided a transformation from an obfuscation of a point function to an obfuscation of a point function with multibit output. This transformation requires

an essential property on the given obfuscation, specifically, composition. In other words, our construction assumes that we have an obfuscation of a point function such that security is not compromised when multiple obfuscated functions are given. Notably, Theorems 8.2.1, 8.2.2, and 8.2.3 all assume that H satisfies some form of composable security. Since the obfuscator is probabilistic, composable security is nontrivial. In this section, we address the question: does the basic definition of obfuscation imply composition? From a different angle, Canetti *et al.* [CMR98] ask if semantic perfect one-wayness implies indistinguishable perfect one-wayness or if t-indistinguishable POW functions are $t+1$-indistinguishable. We answer these questions negatively: such primitives are not necessarily secure even under self-composition.[3] In more detail, we show that weak c-indistinguishable POW functions (where the probability is taken over the choice of the seed as well, refer to [CMR98] or Definition 3.4.2) are not necessarily $(c+1)$-indistinguishable for any constant c. We also show that POW functions, POW functions with auxiliary input, and obfuscation of point functions do not imply composition. Specifically, 1-indistinguishable POW functions and obfuscation of point functions are not necessarily secure for a polynomial number of copies. Moreover, even though 1-indistinguishable POW functions with auxiliary input are also c-indistinguishable for any constant c, they are not necessarily t-indistinguishable with auxiliary input for a polylogarithmic t.

In Section 8.3.1, we show a tight impossibility result for weak POW functions. Specifically, we show that for any constant c, weak c-indistinguishable POW functions are not weakly $(c+1)$-indistinguishable. We also show that if t is a polynomial, then weak t-indistinguishable POW functions are not weakly $n(t+1)$-indistinguishable. In Section 8.3.2, we prove that 1-indistinguishable POW functions and point function obfuscation are not secure if composed roughly $nlog(n)$ times. Moreover, if we consider the same functions with respect to auxiliary information, then we have a tighter result where they are not secure with respect to auxiliary information if composed superlogarithmically-many times.

[3] Recall, self-composition refers to concatenation of multiple outputs of a randomized function on the *same* input.

8.3.1 Weak POW Functions are not Self-composable

Recall from Section 3.4, a weak POW function deviates from Definition 2.5.5 in that the probability is taken over the choice of the function key as well. Here, we show that a weak c-indistinguishable POW function with respect to the uniform distribution may not be $c+1$ indistinguishable for any constant c. The idea is simple: we take any weak $2c$-indistinguishable POW function and convert it into a new function that is c-indistinguishable but the output contains shares of the input such that it is easy to compute the input from $c+1$ images. Informally, we add c uniform strings to the original seed and make sure that an image of the input using any one of these c strings appears in the output with probability $\frac{1}{c+1}$. Also, with the same probability the exclusive-or of the input and all the aforementioned images appears in the output. Therefore, if the output of the function contains all c images and the exclusive-or of these images with the input, then it is easy to recover the input. Formally,

Construction 8.3.1. *Let $\mathbb{H}$ be any (possibly weak) $2c$-pseudorandom POW function with key space, K_n, and public randomness. Define a new family ensemble, $\mathbb{G}$, with a key space $(K_n, \underbrace{R_n, ..., R_n}_{c})$, an input domain $(\{0,1\}^n, \{0,1\}^n)$, and randomness domain $(R_n, \{0,1\}^{logc})$, as follows:*

$$G_{k,u_1,...,u_c}((x_1,x_2),(r_1,r_2)) =$$

$$\begin{cases} r_2, H_k(x_1,r_1), H_k(x_2,r_1), H_k(x_1,u_{r_2}) & \text{if } r_2 \neq 0 \\ r_2, H_k(x_1,r_1), H_k(x_1,u_1) \oplus H_k(x_1,u_2) ... \oplus H_k(x_1,u_c) \oplus x_2 & \text{if } r_2 = 0 \end{cases}$$

Theorem 8.3.1. *If there exist a constant c and any weak POW function that is $2c$-indistinguishable from uniform (as in Definition 2.5.6) and has public randomness, then, there exist weak POW functions that are c-indistinguishable with respect to the uniform distribution but not $(c+1)$-indistinguishable with respect to the uniform distribution.*

Proof. For any weak $2c$-indistinguishable POW function $\mathbb{H}$, apply Construction 8.3.1 on $\mathbb{H}$ to get $\mathbb{G}$.

$\mathbb{G}$ is not weak $(c+1)$-indistinguishable from uniform. Observe that it is easy to recover x_2 from $G_{k,u_1,...,u_c}((x_1,x_2),(r_1^0,0)), ...,$

$G_{k,u_1,...,u_c}((x_1,x_2),(r_1^c,c))$. Thus, for a constant c, $\mathbb{G}$ is not $(c+1)$-indistinguishable

because $c+1$ randomly-chosen images of (x_1, x_2) have distinct r_2 (i.e., match the aforementioned set of images) with probability $\frac{(c+1)!}{(c+1)^{c+1}}$ (a nonnegligible probability).

$\mathbb{G}$ is weak c-indistinguishable from uniform. First, we argue that by the $2c$-indistinguishability of $\mathbb{H}$, for any PPT, A:

$$|Pr[k, u_1, ..., u_c \leftarrow K_n, R_1, ..., R_n,\ x_1, x_2 \leftarrow U_n, U_n,$$

$$(r_1^1, r_2^1), ..., (r_1^c, r_2^c) \leftarrow (R_n, \{0,1\}^{logc}), ..., (R_n, \{0,1\}^{logc}),$$

$$b \leftarrow A(G_{k,u_1,...,u_c}((x_1, x_2), (r_1^1, r_2^1)), ..., G_{k,u_1,...,u_c}((x_1, x_2), (r_1^c, r_2^c))) : b = 1] -$$

$$Pr[k, u_1, ..., u_c \leftarrow K_n, R_1, ..., R_n,\ v_1, ..., v_c, x_2 \leftarrow U_n, ..., U_n,$$

$$(r_1^1, r_2^1), ..., (r_1^c, r_2^c) \leftarrow (R_n, \{0,1\}^{logc}), ..., (R_n, \{0,1\}^{logc}),$$

$$b \leftarrow A(G_{k,u_1,...,u_c}((v_1, x_2), (r_1^1, r_2^1)), ..., G_{k,u_1,...,u_c}((v_c, x_2), (r_1^c, r_2^c))) : b = 1]| \leq \mu(n). \tag{8.6}$$

Suppose, for the purpose of contradiction, there is an adversary, A that contradicts this claim. Let B be an adversary that defeats $2c$-pseudorandomness of $\mathbb{H}$. Then, B receives $2c$ images under $\mathbb{H}$, $y_1, ..., y_{2c}$. B samples x_2 and $r_2^1, ..., r_2^c$ uniformly. It then runs A on $k, r_{c+1}, ..., r_{2c}, w_1, ..., w_c$, where r_i is the public randomness of y_i, and

$$w_i = \begin{cases} r_2^i, y_i, H_k(x_2, r_i), y_{c+r_2^i} & \text{if } r_2 \neq 0 \\ r_2^i, y_i, y_{c+1} \oplus ... \oplus y_{2c} \oplus x_2 & \text{if } r_2 = 0 \end{cases}$$

Note that if the input to B consists of images of the same input then the input given to A is the same as in the first experiment of Eq. 8.6. Moreover, if the input to B consists of images of uniform and independent input, then the input of A is the same as in the second experiment of Eq. 8.6. Thus, B breaks $2c$-indistinguishability of $\mathbb{H}$ with the same probability that A defeats Eq. 8.6, which is assumed to be nonnegligible. This contradicts $2c$-indistinguishability of $\mathbb{H}$.

We then use $2c$-indistinguishability again (this time indistinguishability from uniform) to show that for any PPT, A:

$$|Pr[k, u_1, ..., u_c \leftarrow K_n, R_1, ..., R_n,\ v_1, ..., v_c, x_2 \leftarrow U_n, ..., U_n,$$

$$(r_1^1, r_2^1), ..., (r_1^c, r_2^c) \leftarrow (R_n, \{0,1\}^{logc}), ..., (R_n, \{0,1\}^{logc}),$$

$$b \leftarrow A(G_{k,u_1,...,u_c}((v_1, x_2), (r_1^1, r_2^1)), ..., G_{k,u_1,...,u_c}((v_c, x_2), (r_1^c, r_2^c))) : b = 1] -$$

$$|Pr[k, u_1, ..., u_c \leftarrow K_n, R_1, ..., R_n,\ v_1, ..., v_c, z_1, ..., z_c \leftarrow U_n, ..., U_n,$$

$$(r_1^1, r_2^1), ..., (r_1^c, r_2^c) \leftarrow (R_n, \{0,1\}^{logc}), ..., (R_n, \{0,1\}^{logc}),$$

$$b \leftarrow A(G_{k,u_1,...,u_c}((v_1, z_1), (r_1^1, r_2^1)), ..., G_{k,u_1,...,u_c}((v_c, z_c), (r_1^c, r_2^c))) : b = 1]| \leq \mu(n). \tag{8.7}$$

Again, suppose for the purpose of contradiction, that there is a PPT, A, that contradicts this claim. Construct another PPT, B that uses A to break c-indistinguishability of $\mathbb{H}$. B receives c images, $y_1, ..., y_c$. It chooses $r_2^1, ..., r_2^c$, $v_1, ..., v_c$, and $u_1, ..., u_c$ and d uniformly and runs A on $k, r_1, ..., r_c, w_1, ..., w_c$, where r_i is the public randomness of y_i, and

$$w_i = \begin{cases} r_2^i, H_k(v_i, r_i), y_i, H_k(v_i, u_{r_2^i}) & \text{if } r_2 \neq 0 \\ r_2^i, H_k(v_i, r_i), d & \text{if } r_2 = 0 \end{cases}$$

We argue that if the input to B consists of images of the same point, then the corresponding input to A is computationally indistinguishable from its input in the first experiment of Eq. 8.7. If this were not to be the case, then it is possible to distinguish $2c$ images under $\mathbb{H}$ from uniform (by taking $2c$ hashes as input, choosing x_2 at random, and computing the corresponding distribution). By the same argument, if the input to B consists of images of uniform and independent point, then the corresponding input to A is computationally indistinguishable from its input in the second experiment of Eq. 8.7. Thus, B breaks indistinguishability of $\mathbb{H}$ with nonnegligible probability. A contradiction.

Combining Eq. 8.6 and Eq. 8.7 finishes the proof. □

Moreover, this result can be generalized to any polynomial t. If $\mathbb{H}$ is $2t$-indistinguishable from uniform, then $\mathbb{G}$ is a weak t-indistinguishable POW function with respect to the uniform distribution. On the other hand, $\mathbb{G}$ is not $n(t+1)$-indistinguishable with respect to the uniform distribution. This is so because all the $(t+1)$ "shares" appear in $n(t+1)$ images with overwhelming probability. This result is stated formally in the following theorem.

Theorem 8.3.2. *If there exists a polynomial t and a weak POW function that is $2t$-*

indistinguishable from uniform (as in Definition 2.5.6) and has public randomness, then, there exist weak POW functions that are t-indistinguishable with respect to the uniform distribution but not $n(t+1)$-indistinguishable with respect to the uniform distribution.

Proof. For any weak $2t$-indistinguishable POW function $\mathbb{H}$, apply Construction 8.3.1 on $\mathbb{H}$ to get $\mathbb{G}$.

$\mathbb{G}$ is not weak $n(t+1)$-indistinguishable from uniform. If the $t+1$ shares appear with nonnegligible probability in $n(t+1)$ images then $\mathbb{G}$ is not weak $n(t+1)$-indistinguishable from uniform. The probability that a particular share does not appear in $n(t+1)$ uniformly sampled shares is $(\frac{t}{t+1})^{n(t+1)}$. By the union bound, the probability that a share does not appear in $n(t+1)$ images is at most $(t+1)(\frac{t}{t+1})^{n(t+1)} \leq (t+1)(1-\frac{1}{t+1})^{n(t+1)} \leq (t+1)e^{-n}$. Thus, all $(t+1)$ shares appear in $n(t+1)$ images with high probability.

$\mathbb{G}$ is weak t-indistinguishable from uniform. This proof is exactly the same as the corresponding one in Theorem 8.3.1. □

8.3.2 Point Function Obfuscation and POW Functions Are Not Self-composable

We show that POW functions, POW functions with auxiliary input, obfuscation of point functions, and obfuscation of point functions with auxiliary input are not generally self-composable. Also, we note that the obfuscation of point functions in [Wee05] is not self-composable as well. The idea is simple, we start with a POW function and append to its output a hardcore bit, specifically the inner product between the input and a random string. This hardcore bit does not compromise security of a single image. However, the function becomes completely insecure for polynomially many images as the input can be recovered with high probability by solving a linear system of equations. The results are stated formally as follows.

Construction 8.3.2. *Let $\mathbb{H}$ be a POW function (respectively, point function obfuscation). Define a new family ensemble, $\mathbb{G}$:*

$$G_k(x,(r_1,r_2)) = r_2, H_k(x,r_1), \langle x, r_2 \rangle,$$

where $\langle x, r_2 \rangle$ is the inner product of x and r_2 mod 2.

Theorem 8.3.3. *If there exists a* 1*-indistinguishable POW function (respectively, a point function obfuscation) with auxiliary input then there exists another* 1*-indistinguishable POW function (respectively, another point function obfuscation) with auxiliary input such that for any constants* c *and* ϵ*, the latter is not* t*-indistinguishable (respectively, is not a* t*-self-composable point function obfuscation) with auxiliary input with respect to the uniform distribution , where* $t = \omega(1)log(n)log\frac{\omega(1)log(n)}{-ln(\frac{1}{n^c}+\epsilon)}$.

Moreover, if there exists a 1*-indistinguishable POW function (respectively, a point function obfuscation) then there exists another* 1*-indistinguishable POW function (respectively, another point function obfuscation) such that for any constants* c *and* ϵ*, the latter is not* t*-indistinguishable (respectively, is not a* t*-self-composable point function obfuscation) with respect to the uniform distribution, where* $t = nlog\frac{n}{-ln(\frac{1}{n^c}+\epsilon)}$.

Proof. Here, we prove the result for POW functions with auxiliary input only. The results for the other classes are very similar.

$\mathbb{G}$ is 1-indistinguishable with auxiliary input. For any uninvertible function F, $F(x), H(x, r_1), r_2$ is one-way in x because $\mathbb{H}$ is 1-indistinguishable with auxiliary input. Therefore, by the Goldreich-Levin theorem [GL89], we have that $F(x), r_2, H(x, r_1), \langle x, r_2\rangle$ is indistinguishable from $F(x), r_2, H(x, r_1), b$, where b is uniform. Moreover, by 1-indistinguishability with auxiliary input on $\mathbb{H}$, $F(x), r_2, H(x, r_1), b$, is indistinguishable from $F(x), r_2, H(U_n, r_1), b$.

$\mathbb{G}$ is not polylogarithmically-indistinguishable with auxiliary input. We argue that $\mathbb{G}$ is breakable with respect to the uniform distribution in the presence of polylogarithmic number of images and some auxiliary information. Specifically, let F be a function that outputs the last $n - \omega(1)log(n)$ bits of its input. Then, F is uninvertible with respect to the uniform distribution. However, given $F(x)$ and a t number of images, x can be recovered completely by solving a system of linear equations. Formally,

Lemma 8.3.1. *For any two constants* c *and* ϵ*, there exists a* t*, which is polylogarithmic in* n *(specifically,* $t = \omega(1)log(n)log\frac{\omega(1)log(n)}{-ln(\frac{1}{n^c}+\epsilon)}$*) and a PPT,* A*, such that for any* $k \in K_n$*:*

$$Pr[x \leftarrow U_n,\ r_1, ..., r_t \leftarrow R_n^G, ..., R_n^G,\ A(F(x), G_k(x, r_1), ..., G_k(x, r_t))] \geq \frac{1}{n^c}.$$

Proof. Let A be a PPT that ignores all $\mathbb{H}$ images ($H_k(x, .)$) but plugs-in the values of the last $n - \omega(1)log(n)$ bits of x in the system of linear equations: $r_1^2, \langle x, r_1^2\rangle, ..., r_t^2, \langle x, r_t^2\rangle$.

We show that by solving this system we can recover x with probability $\frac{1}{n^c}$. Given the last $n - \omega(1)log(n)$ bits of x revealed by F, we can recover x from $\omega(1)log(n)$ linearly independent equations on the first $\omega(1)log(n)$ bits. Thus, we need to show that we have this many linearly independent equations in t uniformly chosen equations with probability $\frac{1}{n^c}$. First, observe that a uniform and independent r is linearly independent from $\omega(1)log(n) - 1$ or less equations with probability at least $\frac{1}{2}$. Consequently, the probability that t equations contain $\omega(1)log(n)$ linearly independent equations is at least:

$$\left(1 - \frac{1}{2^{log\frac{\omega(1)log(n)}{-ln(\frac{1}{n^c}+\epsilon)}}}\right)^{\omega(1)log(n)} \geq e^{ln(\frac{1}{n^c}+\epsilon)} - \epsilon = \frac{1}{n^c}.$$

□

□

As a concrete example, note that the main obfuscation of point functions in [Wee05] outputs $\langle x, r\rangle$ in the clear, where x is the point on which the function, F_x, outputs 1 and r is uniform. By Theorem 8.3.3, this construction is not secure when composing $t = nlog\frac{n}{-ln(\frac{1}{n^c}+\epsilon)}$ obfuscated copies of the same point function.

8.4 On the Relationship Between Obfuscation of Multibit Point Functions and Symmetric Encryption

It is interesting to note that obfuscation of point functions with multibit output and symmetric encryption are similar. At the conceptual level, they capture the same idea except with a subtle difference. First, both of them satisfy the same correctness property. In particular, an encryption scheme (respectively, obfuscation of point function with multibit output) allows the recovery of the message (respectively, y) given the key (respectively, x). Second, they share similar privacy requirements. An obfuscation hides the special output, y, of the function, $F_{x,y}$ unless x is given. Likewise, a symmetric encryption should ensure the privacy of the message unless the adversary possesses the key. However, the former primitive differs from the latter in that its behavior is defined over all possible input x, while the decryption scheme leaves the behavior undefined on wrong keys. In other words, one may, at least conceptually, think of an obfuscation of

point functions with multibit output as a *special* form of encryption, where wrong keys are promptly detected by the decryption algorithm.

At a more technical level, another difference arises, regarding the assumption on the key distribution. Recall that symmetric encryption requires uniform keys. On the other hand, an obfuscation of point functions with multibit output does not assume anything about the distribution on x. Specifically, it provides a definition of privacy for any x. Thus, casting the former primitive as an encryption scheme, i.e., as $O(F_{key,message})$, gives us an encryption scheme with the same privacy as defined for obfuscation. In other words, any predicate computed from the ciphertext can also be computed by exhaustively searching for the right key to recover the message. Formally,

Definition 8.4.1 (Single-message encryption for any key). *A symmetric encryption scheme, (E, D), satisfies privacy for **any** key if for any nonuniform PPT A, and any polynomial p, there exists a nonuniform PPT S such that for any key, k, any message, m, and sufficiently large n:*

$$|Pr[b \leftarrow A(E(k,m)) : b = 1] - Pr[b \leftarrow S^{F_{k,m}}(1^n) : b = 1]| \leq \frac{1}{p(n)}.$$

Observe that in the special case where the key is uniform or even sampled from a well-spread distribution, Definition 8.4.1 implies that whatever predicate computed from the ciphertext can be computed *without it (and without oracle access to $F_{k,m}$)*. Formally, an encryption scheme satisfying Definition 8.4.1 also satisfies the following privacy property.

Definition 8.4.2 (Single-message encryption with well-spread keys). *A symmetric encryption scheme, (E, D), satisfies privacy for **well-spread** keys if for any nonuniform PPT A, and any polynomial p, there exists a nonuniform PPT S such that for any well-spread distribution, $\mathbb{K} = \{K_n\}_{n\in\mathbb{N}}$, any message m, and sufficiently large n:*

$$|Pr[k \leftarrow K_n,\ b \leftarrow A(E(k,m)) : b = 1] - Pr[b \leftarrow S(1^n) : b = 1]| \leq \frac{1}{p(n)}.$$

Although Definitions 8.4.1 and 8.4.2 consider single-message encryption, encryption of multiple messages can be readily achieved using appropriately composable obfuscation of point functions with multibit output.

8.4.1 Weakness of Definition 8.4.1

It may seem that Definition 8.4.1 captures our intuition that the only way of breaking the encryption scheme is through exhaustively searching for the correct key. However, it turns out that this definition is not strong enough. Specifically, there are encryption schemes that satisfy this definition but reveal the plaintext when the key is taken from a polynomial-size set. For instance, modify any encryption scheme that satisfies Definition 8.4.1 so that it reveals the plaintext when the key is one of the first n lexicographically-ordered keys. The new scheme still satisfies this definition, because the simulator can query the oracle on those n keys to recover the message. However, this scheme does not match our intuitive requirement. This is so because an adversary can, in constant time, output the first bit of the plaintext on the first n keys but the simulator needs $O(n)$ time to do the same. We stress that this weakness is already inherent in the notion of obfuscation, not just in the application to encryption.

Coming up with a realizable definition that captures our intuition about encryption with low-entropy keys is interesting, given the potential applications in password-based encryption. Here, we take a step in this direction. We strengthen Definition 8.4.1 by restricting the number of queries of the simulator to some fixed polynomial in the running time of the adversary and the simulation error. In more detail, for any key, k, the number of queries the simulator makes in the worst case is bounded by a fixed polynomial in the worst-case running-time of the adversary and the simulation error.

Definition 8.4.3 (t-secure encryption). *A symmetric encryption scheme, (E, D), is t-secure if for any nonuniform PPT A, and any polynomial p, there exists a nonuniform PPT S such that for any key, k, any message, m, and sufficiently large n:*

$$|Pr[b \leftarrow A(E(k,m)) : b = 1] - Pr[b \leftarrow S^{F_{k,m}}(1^n) : b = 1]| \leq \frac{1}{p(n)},$$

where S makes at most $t(R_{A,k,m}, n, p)$ queries and $R_{A,k,m}$ is the worst-case running time of A on $E(k,m)$, taken over the coin tosses of A and E.

The definition of obfuscation can also be strengthened in a similar way. Obviously, the smaller t is, the stronger the security guarantee. For instance, if an encryption scheme (respectively, obfuscation) is t-secure then it (respectively, the obfuscator) can not do certain "stupid" things such as outputting the plaintext (respectively, the original

function) in the clear on more than $\frac{nt(|E(.,.)|,n,n)}{n-1}$ keys (respectively, $\frac{nt(|O(.)|,n,n)}{n-1}$ functions). We note that the construction in Section 8.2 satisfies this definition for some specific t. However, the question remains as to how small t can be made.

Bibliography

[Bab85] L. Babai. Trading group theore for randomness. *ACM Symposium on Theory of Computing (STOC)*, 1985.

[Bar01] B. Barak. How to go beyond the black-box simulation barrier. *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 106–115, 2001.

[BCY89] G. Brassard, C. Crťepeau, and M. Yung. Everything in NP can be argued in perfect zero-knowledge in a bounded number of rounds. *Eurocrypt*, 1989.

[BD07] C. Bosley and Y. Dodis. Does privacy require true randomness? *Theory of Cryptography Conference (TCC)*, 2007.

[BF06] A. Boldyreva and M. Fischlin. On the security of OAEP. *AsiaCrypt*, 2006.

[BFM88] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. *ACM Symposium on Theory of Computing (STOC)*, pages 103–112, 1988.

[BG92] M. Bellare and O. Goldreich. On defining proofs of knowledge. *Crypto*, 1992.

[BGI+01] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. *Crypto*, 2001.

[BJY97] M. Bellare, M. Jakobsson, and M. Yung. Round-optimal zero-knowledge arguments based on any one-way function. *Eurocrypt*, 1997.

[BL04] B. Barak and Y. Lindell. Strict polynomial-time in simulation and extraction. *SIAM Journal on Computing*, 2004.

[Bla79] G. R. Blakley. Safeguarding cryptographic keys. *Proceedings of the National Computer Conference*, 48:313–317, 1979.

[Blu86] M. Blum. How to prove a theorem so no one else can claim it. *Proceedings of the International Congress of Mathematicians*, 1986.

[BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13:850–864, 1984.

[BOV03] B. Barak, S. Ong, and S. Vadhan. Derandomization in cryptography. *Crypto*, 2003.

[BP04a] M. Bellare and A. Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. *Crypto*, 2004.

[BP04b] M. Bellare and A. Palacio. Towards plaintext-aware public-key encryption without random oracles. *Asiacrypt*, 2004.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical:a paradigm for designing efficient protocols. *CCS*, 1993.

[BR94] M. Bellare and P. Rogaway. Optimal asymmetric encryption. *EuroCrypt*, 1994.

[Can97] R. Canetti. Towards realizing random oracles:hash functions that hide all partial information. *Crypto*, 1997.

[CD08a] R. Canetti and R. R. Dakdouk. Extractable perfectly one-way functions. *International Colloquim on Automata, Languages and Programming, Track C*, 2008.

[CD08b] R. Canetti and R. R. Dakdouk. Obfuscating point functions with multibit output. *EuroCrypt*, 2008.

[CD09] R. Canetti and R. R. Dakdouk. Towards a theory of extractable functions. *Theory of Cryptography Conference (TCC)*, 2009.

[CDN01] R. Cramer, I. Damgard, and J.B. Nielsen. Multiparty computation from threshold homomorphic encryption. *EuroCrypt*, 2001.

[CGH98] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *ACM Symposium on Theory of Computing (STOC)*, 1998.

[CGMA85] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 383–395, 1985.

[CMR98] R. Canetti, D. Micciancio, and O. Reingold. Perfectly one-way probabilistic hash functions. *ACM Symposium on Theory of Computing (STOC)*, 1998.

[Dam92] I. Damgard. Towards practical public key systems secure against chosen ciphertext attacks. *Crypto*, 1992.

[DDN00] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30, 2000.

[Den06] A. Dent. The cramer-shoup encryption scheme is plaintext aware in the standard model. *Eurocrypt*, 2006.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 1976.

[DS02] Y. Dodis and J. Spencer. On the (non)universality of the one-time pad. *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2002.

[DS05] Y. Dodis and A. Smith. Entropic security and the encryption of high-entropy messages. *Theory of Cryptography Conference (TCC)*, 2005.

[Fel87] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 427–437, 1987.

[(FI93] Federal Information Processing Standard (FIPS). Secure hash standard. *NIST*, FIPS publication 180, 1993.

[FKSW05] A. Futoransky, E. Kargieman, C. Sarraute, and A. Waissbein. Foundations and applications for secure triggers. *eprint, 284*, 2005.

[FLS99] U. Feige, D. Lapidot, and A. Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM J. Comput.*, 29, 1999.

[FOPS01] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA-OAEP is secure under the RSA assumption. *Crypto*, 2001.

[FPM] Firefox password manager. *http://www.firefoxtutor.com/61/securing-firefox-passwords/.*

[FS86] A. Fiat and A. Shamir. How to prove yourself:practical solutions to identification and signature problems. *Crypto*, 1986.

[FS89] U. Feige and A. Shamir. Zero knowledge proofs of knowledge in two rounds. *Crypto*, 1989.

[FS90] U. Feige and A. Shamir. Witness indistinguishability and witness hiding protocols. *ACM Symposium on Theory of Computing (STOC)*, pages 416–426, 1990.

[Get63] E. Gettier. Is justified true belief knowledge? *Analysis*, 23, 1963.

[GK96] O. Goldreich and H. Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 1996.

[GK03] S. Goldwasser and Y. T. Kalai. On the (in)security of the fiat-shamir paradigm. *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2003.

[GK05] S. Goldwasser and Y. T. Kalai. On the impossibility of obfuscation with auxiliary input. *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2005.

[GL89] O. Goldreich and L. Levin. Hard-core predicates for any one-way function. *ACM Symposium on Theory of Computing (STOC)*, 1989.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Science*, 28, 1984.

[GMR85] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. *ACM Symposium on Theory of Computing (STOC)*, 1985.

[GMW86] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *IEEE Symposium on Foundations of Computer Science (FOCS)*, 1986.

[GO94] O. Goldreich and Y. Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 1994.

[Gol01] O. Goldreich. *Foundations of Cryptography.* Cambridge University Press, 2001.

[GOS06] J. Groth, R. Ostrovsky, and A. Sahai. Non-interactive zaps and new techniques for NIZK. *Crypto*, 2006.

[HMLS07] D. Hofheinz, J. Malone-Lee, and M. Stam. Obfuscation for cryptographic purposes. *Theory of Cryptography Conference (TCC)*, 2007.

[HT98] S. Hada and T. Tanaka. On the existence of 3-round zero-knowledge protocols. *Crypto*, 1998.

[HT99] S. Hada and T. Tanaka. On the existence of 3-round zero-knowledge protocols. *eprint*, 1999.

[Imp95] R. Impagliazzo. Hard-core distributions for somewhat hard problems. *IEEE Symposium on Foundations of Computer Science (FOCS)*, 1995.

[Kat03] J. Katz. Efficient and non-malleable proofs of plaintext knowledge and applications. *Eurocrypt*, 2003.

[Lep02] M. Lepinski. On the existence of 3-round zero-knowledge proofs. *M.S. Thesis*, 2002.

[LPS04] B. Lynn, M. Prabhakaran, and A. Sahai. Positive results and techniques for obfuscation. *Eurocrypt*, 2004.

[MP90] J. McInnes and B. Pinkas. On the impossibility of private key cryptography with weakly random keys. *Crypto*, 1990.

[MRH04] U. Maurer, R. Renner, and C. Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. *Theory of Cryptography Conference (TCC)*, 2004.

[Nao03] M. Naor. On cryptographic assumptions and challenges. *Crypto*, pages 96–109, 2003.

[Nie02] J. Nielsen. Separating random oracle proofs from complexity theoretic proofs:the non-committing encryption case. *Crypto*, 2002.

[NY90] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. *ACM Symposium on Theory of Computing (STOC)*, 1990.

[Pha] Phaedo. *Phaedo*.

[Plaa] Plato. *Meno*.

[Plab] Plato. *Theaetetus*.

[PX09] M. Prabhakaran and R. Xue. Statistically hiding sets. *RSA conference, Cryptography-track*, 2009.

[Riv92] R. Rivest. The MD5 message-digest algorithm. *IETF Network Working Group*, RFC 1321, 1992.

[Sah99] A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. *IEEE Symposium on Foundations of Computer Science (FOCS)*, 1999.

[SCO+01] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust non-interactive zero knowledge. *Crypto*, 2001.

[SCP00] A. De Santis, G. Di Crescenzo, and G. Persiano. Necessary and sufficient assumptions for non-interactive zero-knowledge proofs of knowledge for all NP relations. *International Colloquim on Automata, Languages and Programming*, 2000.

[Sha79] A. Shamir. How to share a secret. *Communications of the ACM*, pages 612–613, 1979.

[SP92] A. De Santis and G. Persiano. Zero knowledge proofs of knowledge without interaction. *IEEE Symposium on Foundations of Computer Science (FOCS)*, 1992.

[Wee05] H. Wee. On obfuscating point functions. *ACM Symposium on Theory of Computing (STOC)*, 2005.

[Yao82] A.C. Yao. Theory and application of trapdoor functions. *IEEE Symposium on Foundations of Computer Science (FOCS)*, 1982.

Appendix A

General Definitions of Interactively-extractable Functions

The general form (with an even number of rounds) of the 3-round game described in Chapter 4 starts with the challenger, C, sending random coins, r_1, in the first round, then A sends the response, $H_k(x, r_1)$, in the second round. The remaining rounds follow the same pattern. As in the original formulation, no verification occurs. In particular, A may be sending random strings. The consequences include, ofcourse, the invalidity of the consistency and thus inability of preimage extraction. On the other hand, if A plays the game consistently, then we require an extractor to recover a preimage common to all images sent. The usefulness of this notion is apparent when this game is embedded in a protocol where verification can actually occur, e.g., the Random Oracle instantiation in encryption schemes (see Chapter 7).

In the case of an odd number of rounds, the game starts with A sending an image of x with an r of its choice and then the game proceeds as above. Let $r_1, ..., r_{t(n)}$ denote the list of random coins that C sends and $y_1, ..., y_{t(n)}$ denote the corresponding response of A.

A.1 Preimage Knowledge without Auxiliary Information

Again, there are two notions, one that holds for any function and another for a uniformly chosen one.

Definition A.1.1 ((Strong) Interactive extraction without auxiliary informa-

tion). *A verifiable family ensemble,* $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$*, is called (strongly)* $(2t+1)$***-round extractable without auxiliary information*** *if for any PPT, A (with private random coins denoted by* r_A*), and polynomial, p, there exists a PPT,* $\mathcal{K}_A$ *such that for any* $k \in K_n$*:*

$$Pr[r_1, ..., r_{t(n)} \leftarrow R_n, ..., R_n,$$

$$(y_0, s) = A(k, r_A),\ (y_1, ..., y_{t(n)}) =< A(s, r_A), C(r_1, ..., r_{t(n)}) >,$$

$$x \leftarrow \mathcal{K}_A(k, r_1, ..., r_{t(n)}, r_A) :$$

$$(V_{\mathbb{H}}(x, y_0) = 1 \text{ and } \forall i,\ y_i = H_k(x, r_i)) \text{ or } (\forall x',\ \exists i,\ y_i \neq H_k(x', r_i) \text{ or } V_{\mathbb{H}}(x', y_0) \neq 1)]$$

$$> 1 - \frac{1}{p(n)} - \mu(n).$$

Definition A.1.2 (Interactive extraction without auxiliary information). *A verifiable family ensemble,* $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$*, is called* $(2t+1)$***-round extractable without auxiliary information*** *if for any PPT, A (with private random coins denoted by* r_A*), and polynomial, p, there exists a PPT,* $\mathcal{K}_A$ *such that:*

$$Pr[k \leftarrow K_n,\ r_1, ..., r_{t(n)} \leftarrow R_n, ..., R_n,$$

$$(y_0, s) = A(k, r_A),\ (y_1, ..., y_{t(n)}) =< A(s, r_A), C(r_1, ..., r_{t(n)}) >,$$

$$x \leftarrow \mathcal{K}_A(k, r_1, ..., r_{t(n)}, r_A) :$$

$$(V_{\mathbb{H}}(x, y_0) = 1 \text{ and } \forall i,\ y_i = H_k(x, r_i)) \text{ or } (\forall x',\ \exists i,\ y_i \neq H_k(x', r_i) \text{ or } V_{\mathbb{H}}(x', y_0) \neq 1)]$$

$$> 1 - \frac{1}{p(n)} - \mu(n).$$

The definitions for the even round versions are very similar. Note that the 1-round versions of these definitions correspond directly to noninteractive extraction (Definitions 3.2.1 and 3.2.2), except with noticeable error.

A.2 Preimage Knowledge with Independent Auxiliary Information

Adding auxiliary information to Definition A.1.1 yields a definition for dependent auxiliary information. So, we present this notion in the next section. Here, we add independent auxiliary information to Definition 3.2.2.

Definition A.2.1 (Interactive extraction with independent auxiliary information). *Let $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$ be any family ensemble, where $H_k : \{0,1\}^n \times R_n \to \{0,1\}^{l(n)}$ for some polynomial l. Then, $\mathbb{H}$ is called $(2t+1)$-**round extractable** with independent auxiliary information if for any PPT, A (with private random input, r_A), and polynomial, p, there exists a PPT, $\mathcal{K}_A$, such that for any distribution $\mathbb{Z} = \{Z_n\}_{n\in\mathbb{N}}$:*

$$Pr[k \leftarrow K_n,\ z \leftarrow Z_n,\ r_1, ..., r_{t(n)} \leftarrow R_n, ..., R_n,$$

$$(y_0, s) = A(k, z, r_A),\ (y_1, ..., y_{t(n)}) =< A(s, r_A), C(r_1, ..., r_{t(n)}) >,$$

$$x \leftarrow \mathcal{K}_A(k, z, r_1, ..., r_{t(n)}, r_A) :$$

$$(V_{\mathbb{H}}(x, y_0) = 1 \text{ and } \forall i,\ y_i = H_k(x, r_i)) \text{ or } (\forall x',\ \exists i,\ y_i \neq H_k(x', r_i) \text{ or } V_{\mathbb{H}}(x', y_0) \neq 1)]$$

$$> 1 - \frac{1}{p(n)} - \mu(n).$$

The even round version of this definition is similar. Again, 1-round extractable functions with independent auxiliary information correspond to noninteractive extractable functions with independent auxiliary information except with noticeable extraction error.

A.3 Preimage Knowledge with Dependent Auxiliary Information

Definition A.3.1 (Interactive extraction with dependent auxiliary information). *Let $\mathbb{H} = \{\mathbf{H}^n\}_{n\in\mathbb{N}}$ be any family ensemble, where $H_k : \{0,1\}^n \times R_n \to \{0,1\}^{l(n)}$ for some polynomial l. Then, $\mathbb{H}$ is called $(2t+1)$-**round extractable** with dependent auxiliary information if for any PPT, A (with private random input, r_A), and polynomial, p, there exists a PPT, $\mathcal{K}_A$, such that for any distribution $\mathbb{Z} = \{Z_n\}_{n\in\mathbb{N}}$ and any*

$k \leftarrow K_n$:

$$Pr[z \leftarrow Z_n,\ r_1, ..., r_{t(n)} \leftarrow R_n, ..., R_n,$$

$$(y_0, s) = A(k, z, r_A),\ (y_1, ..., y_{t(n)}) =< A(s, r_A), C(r_1, ..., r_{t(n)}) >,$$

$$x \leftarrow \mathcal{K}_A(k, z, r_1, ..., r_{t(n)}, r_A) :$$

$$(V_{\mathbb{H}}(x, y_0) = 1 \ and \ \forall i,\ y_i = H_k(x, r_i)) \ or \ (\forall x',\ \exists i,\ y_i \neq H_k(x', r_i) \ or \ V_{\mathbb{H}}(x', y_0) \neq 1)]$$

$$> 1 - \frac{1}{p(n)} - \mu(n).$$

From a different angle, these general definitions involve a sequential repetition of a 2-round interaction where the challenger sends a challenge r and the adversary responds with the corresponding answer. Consequently, the 3-round version as described in Chapter 4 considers a parallel version, where the challenger sends all of its challenges in one round.