Building Hard Problems by Combining Easy Ones

Riddhi Ghosal[⋆] and Amit Sahai[⋆⋆]

UCLA

Abstract. In this work, we initiate a new conceptual line of attack on the fundamental question of how to generate *hard problems*. Motivated by the need for one-way functions in cryptography, we propose an information-theoretic framework to study the question of generating new *provably* hard one-way functions by composing functions that are *easy* to invert and evaluate, where each such easy function is modeled as a random oracles paired with another oracle that implements an inverse function.

1 Introduction

The search for computational hardness. The abundance of hard problems has been well-known to us since the seminal work of Shannon [Sha49] which proved that almost all boolean functions have exponential circuit complexity. However, not much progress has been made in answering the fundamental question of proving that a particular boolean function requires exponentially large circuits. In fact, we have not even succeeded in identifying concrete families of functions that require super-linear circuit complexity. At the same time, the *conjectured* hardness of computing specific functions (or satisfying certain relations) is necessary for the field of cryptography for the construction of primitives ranging from psueodo-random generators [BM19] and public-key encryption [GM19] to secure two-party computation [Yao82] and indistinguishability obfuscation [BGI$^+$01,JLS21].

The most basic form of hardness that is necessary and useful for cryptography is called a one-way function [Gol01]: a function f such that: (1) Given any valid input x, it is easy to compute $f(x)$; and (2) Given $f(x)$ for a randomly chosen x in the domain of f, it is hard to compute any x' such that $f(x') = f(x)$. Since hard problems are of great importance in cryptography (see later in this introduction for more elaboration on this), the lack of techniques or "recipes" to build such objects is a major over-arching concern.

Can we make hard one-way functions by combining easy functions? How might we build such a recipe for constructing one-way functions, when we don't even know how to prove super-linear circuit lower bounds? In our work, we propose an information-theoretic framework in which to begin to investigate this question. The basic hypothesis we would like to model and test is the following:

A good way to make hard one-way functions is to combine "unrelated" easy functions.

Why start with "easy" functions? Well, the basic mathematical building blocks we typically encounter are easy in *both* directions: For example, given a linear transformation T, it is easy to both: (1) Given a vector x, compute $y = Tx$; and (2) Given $y = Tx$, find a vector x' such that $Tx' = y$. The same is true for the "add a constant" function or the "multiply modulo a prime p" function. Since these are ingredients we are familiar with, it makes sense to try to start by building a "generic" model for an easy function.

We abstract such an easy function by considering our computational entities as having access to certain functions given as an oracle [Tur39,Pos44,Soa99]. Namely, we provide: (1) a random function $\mathsf{O} : \{0,1\}^n \to \{0,1\}^m$ as one oracle, and (2) the inverse function $\mathsf{O}^{-1} : \{0,1\}^m \to \{0,1\}^n \cup \{\perp\}$ as an oracle as well[1]. Since we provide both O and O^{-1} as oracles, they are easy to compute in both directions – just by asking the

[⋆] Email: `riddhi@cs.ucla.edu`.

[⋆⋆] Email: `sahai@cs.ucla.edu`.

[1] For simplicity in this initial work, we restrict our attention to the case where m is significantly larger than n, as this makes O return responses which have a unique pre-image with overwhelming probability.

oracle to do so. However, by leaving the rest of their structure random, we model the idea that the internal mathematical structure of this "easy" function is unrelated to any other easy function that we might want to use. We stress that the purpose of this paper is to initiate this line of study with a simple formalization of "easiness" – we hope future work will explore other formalizations.

Our Results. In this paper, we initiate this investigation by studying two of perhaps the most natural "recipes" that come to mind.

– **Adding two easy functions:** Our first result, informally speaking, is a demonstration that if $f_1, f_2 : \{0,1\}^n \to \{0,1\}^{n+t}$ are two easily invertible and unrelated functions, then the map $x \mapsto f_1(x) + f_2(x)$ is a one-way function[2]. More specifically, we show that for large enough values of t, then $x \mapsto f_1(x) + f_2(x)$ is almost as hard to invert as a truly random function $R : \{0,1\}^n \to \{0,1\}^{n+t}$.
– **Injecting linear noise to an easy function:** Observe that the result above requires *two* independent unrelated "easy" functions modeled as pairs of oracles. Can we use just one such "easy" function, and combine it with an extremely simple function – like adding a fixed constant over a finite field – and still create a hard-to-invert function? Note that if we were to compose our easy-to-invert function with the function that adds a known constant, then the resulting function would also necessarily be easy to invert. Taking inspiration from the hardness of decoding random linear codes [BFKL94,Reg09], we instead introduce *randomness* in a very simple way to help avoid this roadblock: we will apply the simple add-a-constant function to each portion of the output of the "easy" function, but only with some independent probability for each portion of the output.
  More precisely, we interpret the output of the "easy" function as a k-dimensional vector over $\mathbb{F}_{2^m}$, that is, $f : \{0,1\}^n \to \{0,1\}^{km=n+t} = \mathbb{F}_{2^m}^k$ for appropriate $k \in \mathbb{N}$, i.e., if $f(x) = y$, then we re-write $y = (y_1, \dots, y_k)$, such that each $y_i \in \mathbb{F}_{2^m}$.
  - The resultant function is constructed by adding a fixed error $e \in \mathbb{F}_{2^m}$ where $e \neq 0$ to every component with some probability $p > 0$. In other words every y_i gets transformed to $y_i + e$ with probability p and is unaltered with probability $1 - p$. We show that for appropriate values of k, t and p, this function is one-way.
  - We also extend this to the case where there is fixed error list $\mathbf{E} \subset \mathbb{F}_{2^m}$ not containing 0, of size L. To generate the function, we pick k many error terms randomly from $\mathbf{E}$ with replacement and add it to each of the components. Thus, y_i becomes $y_i + e_i, \forall i$, where $e_i \in \mathbf{E}$. In this case, an error list of size 2, i.e., $L = 2$ suffices to prove one-wayness of the resulting function for appropriate choices of parameters.

  In the above constructions, one-wayness holds even if the errors are adversarially chosen.

A more formal description of our results have been presented in Section 2.

Background on the usefulness of hardness in cryptography. Hardness conjectures (typically called assumptions in the cryptographic literature) are ubiquitous in cryptography. Several well-founded assumptions like factoring and RSA [RSA78,Rab79], discrete-log (DLog) [DH22], Decisional linear (DLin) [BBS04], learning parity with noise (LPN) [Ajt96], learning with errors (LWE) [Reg09], multivariate quadratic (MQ) [FY80], quadratic residuosity [GM19] etc. have been proposed. All of these have led to the construction of several primitives like pseudorandom generators [Reg09,PS98], public key encryption schemes [GM19,GSW13,Ale03], signatures [KPG99,BR93,Lyu12,GPV08], zero-knowledge protocols [GMW87,PS19], functional encryption [ABDCP15,Wee20,BSW11], indistinguishability obfuscation [JLS21], and many more.

However, the number of such well-studied hardness assumptions remain limited, and it seems that remarkably many natural mathematical problems which we encounter are easy to solve in polynomial time. Several proposals of new assumptions have either been broken or their hardness have not been sufficiently scrutinised which makes them undesirable candidates to construct secure schemes, especially with quantum computing [Sho94]. This causes a major bottle-neck, thereby making it further challenging to produce novel protocols whose security rests on solid foundations. This naturally motivates the question of whether it is

[2] Here we can interpret $+$ as addition modulo 2^{n+t} or as addition in $\mathbb{F}_{2^{n+t}}$.

possible to prepare some recipes which could be used as toolkits to produce new problems which will be provably hard. Such a technique combined with some well-known hardness assumptions could provide us with ways to generate novel hard problems from readily available mathematical problems which are better suited to prove security of a particular construction. In this work, we take some first steps towards attacking this fundamental question from an information-theoretic point of view.

The intuition behind the existence of such a technique has been inspired from the Learning With Errors (LWE) assumption. Recall the search LWE assumption states that no polynomial time adversary can extract x upon given access to $(A, f(x) = Ax + e)$ for a uniformly chosen matrix $A \in \mathbb{F}_q^{m \times n}$, uniformly chosen fixed and secret vector $x \in \mathbb{F}_q^n$, and a noise vector $e \in \mathbb{F}_q^m$ from an appropriate discrete gaussian distribution. Now, observe that when $m \geqslant n$, search LWE is essentially a composition of an easy to invert problem with another extremely simple function, namely, $f(x) = g(h(x))$, where $g(x) = x + e$ and $h(x) = Ax$. The system of equations generated in $h(x)$ can be solved efficiently using Gaussian Elimination and $g(x)$ is just an addition function which injects some bounded randomness. However, combining these operations somehow yields a pseudorandom generator which has been a basis for numerous post-quantum public key schemes.

2 Technical Overview

A natural way of proving one-wayness of f is to show that the constructed function f is essentially as hard to invert as a random function R. One way of formally capturing this would be to show that any *computationally unbounded* distinguisher D with oracle acccess to either f or R cannot distinguish between the two cases by making at-most polynomially many oracle queries. Formally, this is denoted by the following statement:

$$\left|\Pr[\mathsf{D}^f(1^n) = 1] - \Pr[\mathsf{D}^R(1^n) = 1]\right| \leqslant \mathsf{negl}(n).$$

In our case f is constructed by combining multiple easy problems $(\mathsf{O}_1, \mathsf{O}_2, \ldots)$ and the “easiness” of these “building blocks” is then abstracted by assuming that everyone has oracle access to the function pairs $(\mathsf{O}_1, \mathsf{O}_1^{-1}, \mathsf{O}_2, \mathsf{O}_2^{-1}, \ldots)$ as mentioned above. We now take a step forward and provide a stronger form of the above *indisguishability* notion. Here, D will have oracle access to $(f, (\mathsf{O}_1, \mathsf{O}_1^{-1}, \mathsf{O}_2, \mathsf{O}_2^{-1}, \ldots))$ in the real world. In the ideal world, we introduce an efficient algorithm Sim that interacts with R (denoted by Sim^R) to “simulate” the “easy problems” and provide D with oracle access to (R, Sim^R). In particular, we show that for any distinguisher D making polynomial many queries the following must hold,

$$\left|\Pr[\mathsf{D}^{(f,(\mathsf{O}_1,\mathsf{O}_1^{-1},\mathsf{O}_2,\mathsf{O}_2^{-1},\ldots))}(1^n) = 1] - \Pr[\mathsf{D}^{R,\mathsf{Sim}^R}(1^n) = 1]\right| \leqslant \mathsf{negl}(n).$$

This is the notion of *indifferentiability* introduced in [MRH04].

1. **Sum of two invertible ideal functions.** $f(x) = ((\mathsf{O}_1(x) + \mathsf{O}_2(x)) \mod 2^m)$ with O_1 and O_2 being random functions from n to $m = n + t(n)$ bits. We show that there exists an efficient and deterministic simulator Sim with oracle access to a random function $\mathsf{R} : \{0,1\}^n \to \{0,1\}^m$ (denoted by Sim^R) such that for all distinguishers D making at-most polynomial many oracle queries, the following two worlds where D has oracle access to (1) $(f, (\mathsf{O}_1, \mathsf{O}_1^{-1}, \mathsf{O}_2, \mathsf{O}_2^{-1}))$ vs. (2) $(\mathsf{R}, \mathsf{Sim}^\mathsf{R})$ are distinguishable with a negligible probability. Here is the formal theorem statement:

   Theorem 1. *Let $\mathsf{O}_1, \mathsf{O}_2, \mathsf{R}$ are random functions from $\{0,1\}^n$ to $\{0,1\}^m$, $m = n + t(n)$, $m, n, t \in \mathbb{N}$. For all polynomial-query-bounded distinguishers D making at most $q = \mathsf{poly}(n)$ queries to each oracle, there exists a poly-time oracle simulator $\mathsf{Sim}^{(\cdot)}$ and a positive constant c such that*

   $$\left|\Pr\left[\mathsf{D}^{(\mathsf{O}_1+\mathsf{O}_2),(\mathsf{O}_1,\mathsf{O}_2,\mathsf{O}_1^{-1},\mathsf{O}_2^{-1})}(1^n) = 1\right] - \Pr\left[\mathsf{D}^{\mathsf{R},\mathsf{Sim}^\mathsf{R}}(1^n) = 1\right]\right| \leqslant \frac{cq}{2^t} + \frac{6q^2}{2^{n+t}}.$$

   Theorem 4 shows this result can also be extended to the case where O_1 and O_2 are random permutations, i.e., $\mathsf{O}_i : \{0,1\}^n \to \{0,1\}^n$ is a random bijective function and $\mathsf{O}_i^{-1} : \{0,1\}^n \to \{0,1\}^n$, for all $i \in \{1,2\}$. Refer to Section 4.2 for details.

2. **Injecting Additive Noise.**
   - We assume the existence of a fixed and public (possibly chosen adversarially) list of errors $\mathbf{E}$ of length $L \geqslant 2$ where every element of $\mathbf{E}$ belongs to $\mathbb{F}_{2^m}$. This is added to the output of an easy to invert function. Assume $\mathsf{O} : \mathbb{F}_{2^n} \to \mathbb{F}_{2^m}^k$ for some $k \in \mathbb{N}$ and $km = n + t$. Our construction works as follows: Sample $e_1, \ldots, e_k$ uniformly at random from $\mathbf{E}$ and interpret $\mathsf{O}(x)$ as $(\mathsf{O}(x)^{(1)}, \ldots, \mathsf{O}(x)^{(k)})$, where $\mathsf{O}(x)^{(i)} \in \mathbb{F}_{2^m}$. Define $f(x) = \left((\mathsf{O}(x)^{(1)} + e_1) \mod 2^m\right), \ldots, \left((\mathsf{O}(x)^{(k)} + e_k) \mod 2^m\right)$.
   - Here we assume that there exists an adversarially chosen error $e \in \mathbb{F}_{2^m}$. $\mathsf{O} : \mathbb{F}_{2^n} \to \mathbb{F}_{2^m}^k$ for some $k \in \mathbb{N}$ and $km = n + t$. To construct our function, we first sample bits $b_1, \ldots, b_k$ from a bernoulli distribution with parameter p. Interpret $\mathsf{O}(x)$ as $(\mathsf{O}(x)^{(1)}, \ldots, \mathsf{O}(x)^{(k)}$, where $\mathsf{O}(x)^{(i)} \in \mathbb{F}_{2^m}$ and define $f(x) = \left((\mathsf{O}(x)^{(1)} + b_i \cdot e) \mod 2^m\right), \ldots, \left((\mathsf{O}(x)^{(k)} + b_k \cdot e) \mod 2^m\right)$.

   In both the above, we show that there exists an efficient simulator Sim such that any distinguisher D making polynomially many queries cannot distinguish between $(f, (\mathsf{O}, \mathsf{O}^{-1}))$ and (R, Sim^R) where R is a random function from n to km bits. Below, we present a general theorem which captures both the constructions mentioned above.

   Theorem 2. *Let p be the parameter of a Bernoulli random variable, $k(n) \in \mathbb{N}, m(n), t(n)$ be functions of n. Let $\mathsf{O} : \mathbb{F}_{2^n} \to \mathbb{F}_{2^m}^k$ and $\mathsf{R} : \mathbb{F}_{2^n} \to \mathbb{F}_{2^m}^k$ be random functions such that $km = n + t$, $kp \geqslant c_1 t$ and $k - kp \geqslant c_2 t$, for some constants $c, c_1, c_2 > 0$. For all distinguishers $\mathsf{D} := (\mathsf{D}_1, \mathsf{D}_2)$ making at most $q = \mathsf{poly}(n)$ queries to each oracle, there exists a poly-time oracle simulator $\mathsf{Sim}^{(\cdot)}$ and constant $\alpha > 0$, such that*
   - *when $L \geqslant 2, p = 1$, we have*

   $$\left|\Pr\left[\mathsf{D}_2^{f,(\mathsf{O},\mathsf{O}^{-1})}(1^n, \mathbf{E}) = 1 \middle| \mathbf{E} \leftarrow \mathsf{D}_1(1^n, L)\right] - \Pr\left[\mathsf{D}_2^{\mathsf{R},\mathsf{Sim}^{\mathsf{R}}}(1^n, \mathbf{E}) = 1 \middle| \mathbf{E} \leftarrow \mathsf{D}_1(1^n, L)\right]\right| \leqslant \frac{cq}{2^t} + \frac{q}{L^{c_1 t}},$$

   - *and when $L \geqslant 1, 0 < p < 1$, we have*

   $$\left|\Pr\left[\mathsf{D}_2^{f,(\mathsf{O},\mathsf{O}^{-1})}(1^n, \mathbf{E}) = 1 \middle| \mathbf{E} \leftarrow \mathsf{D}_1(1^n, L)\right] - \Pr\left[\mathsf{D}_2^{\mathsf{R},\mathsf{Sim}^{\mathsf{R}}}(1^n, \mathbf{E}) = 1 \middle| \mathbf{E} \leftarrow \mathsf{D}_1(1^n, L)\right]\right| \leqslant \frac{cq}{2^t} + \frac{\alpha q(k+1)}{e^{c_2 t}}.$$

   Here f is the function defined above.

In the above constructions, we note that the distinguishing probability becomes negligible when $t = \Omega(\log^{1+\varepsilon} n)$.

3 Preliminaries

3.1 Definitions

Definition 1 (Polynomial-Query-Bounded Oracle Turing Machine). *We say that an oracle Turing machine $\mathsf{T}^{(\cdot)}$ is polynomial-query-bounded if there exists a polynomial $p(\cdot) : \mathbb{N} \to \mathbb{N}$ such that for any input $x \in \{0,1\}^*$ and for any oracle O, the execution of $\mathsf{T}^{\mathsf{O}}(x)$ makes at most $p(|x|)$ many queries to O.*

Definition 2 (Indifferentiability [MRH04,DKT16]). *Let $\mathbf{C} : \{0,1\}^n \to \{0,1\}^{m(n)}$ be a construction which has access to an ideal primitive $\mathsf{F} : \{0,1\}^{p(n)} \to \{0,1\}^{q(n)}$ and implements a functionality based on F, where $p(n), q(n), m(n) = \mathsf{poly}(n)$. We say that $\mathbf{C}$ is* indifferentiable *from a random function $\mathsf{RO} : \{0,1\}^n \to \{0,1\}^m$, if there exists a poly-time simulator Sim with oracle access to RO such that for all polynomial-query-bounded distinguishers D ,*

$$\left|\Pr\left[\mathsf{D}^{\mathbf{C}^{\mathsf{F}},\mathsf{F}}(1^n) = 1\right] - \Pr\left[\mathsf{D}^{\mathsf{RO},\mathsf{Sim}^{\mathsf{RO}}}(1^n) = 1\right]\right|$$

is negligible.

4 Construction based on Sum of Random Oracles

Let $\mathsf{O}_1 : \{0,1\}^n \to \{0,1\}^m$ and $\mathsf{O}_2 : \{0,1\}^n \to \{0,1\}^m$, $m = n + t$ be two random oracles. We claim that the function $((\mathsf{O}_1 + \mathsf{O}_2) \mod 2^m)$ is *indifferentiable* from a random function upon giving oracle access to $(\mathsf{O}_1, \mathsf{O}_2, \mathsf{O}_1^{-1}, \mathsf{O}_2^{-1})$, where O_1^{-1} and O_2^{-1} denote the corresponding inverse oracles. We split our results into two cases, (1) O_1 and O_2 are random functions, and (2) O_1 and O_2 are random permutations. In the former scenario, the inverse oracle returns a $\perp$ if queried on an instance which does not have a defined pre-image. For notational simplicity, we drop the modulus operation when it is clear from context.

Notation: Let Reg be a data-structure that stores a list of tuples $(a \in \{0,1\}^n, b \in \{0,1\}^m)$ which is dynamically built on the fly. From here on, in the entire paper, we use $\mathsf{Reg}(a) = b$ to denote $(a, b) \in \mathsf{Reg}$. Similarly, " set $\mathsf{Reg}(x) = y$" simply means adding the pair (x, y) to Reg. This is just for notational simplicity and does not imply that Reg has a space-complexity of 2^n. $\mathsf{BotList}$ denotes the list of all inverse queries that receive a $\perp$ response. ImList denotes another list of all responses sent as response to a forward query of the form $\mathsf{O}(a)$.

4.1 Adding Random Functions

Formally, the inverse oracles are defined as $\mathsf{O}_1^{-1} : \{0,1\}^m \to \{0,1\}^n \cup \{\perp\}$, and $\mathsf{O}_2^{-1} : \{0,1\}^m \to \{0,1\}^n \cup \{\perp\}$.

Theorem 3. *Let $\mathsf{O}_1, \mathsf{O}_2, \mathsf{R}$ are random functions from $\{0,1\}^n$ to $\{0,1\}^m$, $m = n + t(n)$, $m, n, t \in \mathbb{N}$. For all polynomial-query-bounded distinguishers D making at most $q = \mathsf{poly}(n)$ queries to each oracle, there exists a poly-time oracle simulator $\mathsf{Sim}^{(\cdot)}$ and a constant $c > 0$ such that*

$$\left|\Pr\left[\mathsf{D}^{(\mathsf{O}_1+\mathsf{O}_2),(\mathsf{O}_1,\mathsf{O}_2,\mathsf{O}_1^{-1},\mathsf{O}_2^{-1})}(1^n) = 1\right] - \Pr\left[\mathsf{D}^{\mathsf{R},\mathsf{Sim}^{\mathsf{R}}}(1^n) = 1\right]\right| \leqslant \frac{cq}{2^t} + \frac{6q^2}{2^{n+t}}.$$

Note that D's advantage becomes negligible when $t = \Omega(\log^{1+\varepsilon} n), \varepsilon > 0$

Proof. We prove the result using a sequence of intermediate indifferentiable hybrids. For notational simplicity, we use $(\mathsf{LOra}_i, \mathsf{ROra}_i)$ to denote the two oracles accessed by D in Hybrid i.

- **Hybrid** H_1: This represents the case where D interacts with oracles $(\mathsf{R}, \mathsf{Sim}^{\mathsf{R}})$. Hence,
  - LOra_1 is simply R which on input an n bit binary string outputs a random element from $\{0,1\}^n$.
  - ROra_1 is identical to $\mathsf{Sim}^{\mathsf{R}}$ and is defined as follows:

```
∗ On queries of the form $\mathsf{O}_1(a)$
    · if $\mathsf{Reg}_1(a)$ is undefined
      1. $\mathsf{Reg}_1(a) \xleftarrow{\$} \{0,1\}^m \backslash \mathsf{BotList}_1$.
      2. if $\mathsf{Reg}_1(a) \in \mathsf{ImList}_1$, then abort
      3. $\mathsf{ImList}_1 = \mathsf{ImList}_1 \cup \{\mathsf{Reg}_1(a)\}$
      4. Send $a$ as a query to $\mathsf{R}$.
      5. if $\mathsf{R}(a) - \mathsf{Reg}_1(a) \in \mathsf{ImList}_2$, then abort
      6. Set $\mathsf{Reg}_2(a) = \mathsf{R}(a) - \mathsf{Reg}_1(a)$.
      7. $\mathsf{ImList}_2 = \mathsf{ImList}_2 \cup \{\mathsf{Reg}_2(a)\}$
    · return $\mathsf{Reg}_1(a)$.

∗ On queries of the form $\mathsf{O}_1^{-1}(b)$
  1. if $\nexists x$ st $\mathsf{Reg}_1(x) = b$.
    (a) $\mathsf{BotList}_1 \leftarrow \mathsf{BotList}_1 \cup \{b\}$.
    (b) return $\perp$
  2. return $x$ such that $\mathsf{Reg}_1(x) = b$.

∗ On queries of the form $\mathsf{O}_2(a)$
    · if $\mathsf{Reg}_2(a)$ is undefined
      1. $\mathsf{Reg}_2(a) \xleftarrow{\$} \{0,1\}^m \backslash \mathsf{BotList}_2$.
      2. if $\mathsf{Reg}_2(a) \in \mathsf{ImList}_2$, then abort
      3. $\mathsf{ImList}_2 = \mathsf{ImList}_2 \cup \{\mathsf{Reg}_2(a)\}$
      4. Send $a$ as a query to $\mathsf{R}$.
      5. if $\mathsf{R}(a) - \mathsf{Reg}_2(a) \in \mathsf{ImList}_1$, then abort
      6. Set $\mathsf{Reg}_1(a) = \mathsf{R}(a) - \mathsf{Reg}_2(a)$.
      7. $\mathsf{ImList}_1 = \mathsf{ImList}_1 \cup \{\mathsf{Reg}_1(a)\}$
    · return $\mathsf{Reg}_2(a)$.

∗ On queries of the form $\mathsf{O}_2^{-1}(b)$
  1. if $\nexists x$ st $\mathsf{Reg}_2(x) = b$.
    (a) $\mathsf{BotList}_2 \leftarrow \mathsf{BotList}_2 \cup \{b\}$.
    (b) return $\perp$
  2. return $x$ such that $\mathsf{Reg}_1(x) = b$.
```

By definition, we have that

$$\Pr\left[\mathsf{D}^{\mathsf{R},\mathsf{Sim}^{\mathsf{R}}}=1\right]=\Pr\left[\mathsf{D}^{\mathsf{H}_1}=1\right]$$

– **Hybrid** H_2:
  - LOra_2 acts as a forwarding polynomial time-oracle Turing Machine which upon input $a \in \{0,1\}^n$ forwards queries to R and outputs its response.
  - ROra_2 remains identical to ROra_1.

  Thus,

$$\Pr\left[\mathsf{D}^{\mathsf{H}_2}=1\right]=\Pr\left[\mathsf{D}^{\mathsf{H}_1}=1\right]$$

.

– **Hybrid** H_3:
  - LOra_3 has oracle access to ROra_3 and upon queried $a \in \{0,1\}^n$:
    1. Forward any query a as a forward oracle queries for both O_1 and O_2 to ROra_3. Say the responses are b_1, b_2 respectively.
    2. Return $b_1 + b_2$.
  - ROra_3 remains identical as ROra_2.

  Note that the view of D remains unchanged from outcomes of LOra_2 and LOra_3 as the outputs are identically distributed given the abort conditions in Hybrid H_2 are not triggered. Now, the probability that any aborts happens is at most $\frac{q^2}{2^m}$. This is because, at any time there are no more than q items in $\mathsf{ImList_i}$, and the probability that a randomly chosen element in $\{0,1\}^m$ collides with a fixed element in the image list is exactly $\frac{1}{2^m}$.
  Hence,

$$\left|\Pr\left[\mathsf{D}^{\mathsf{H}_3}=1\right]-\Pr\left[\mathsf{D}^{\mathsf{H}_2}=1\right]\right| \leqslant \frac{4q^2}{2^m}.$$

– **Hybrid** H_4:
  - LOra_4 remains identical as LOra_3.
  - ROra_4 stores an additional register OS and no longer has oracle access to R. Rather, it lazy-samples values uniformly at random from $\{0,1\}^m$ locally as responses to the forward oracle queries. Formally it is defined as follows:

```
* On queries of the form $O_1(a)$
  · if $Reg_1(a)$ is undefined
    1. $Reg_1(a) \xleftarrow{\$} \{0,1\}^m \setminus BotList_1$.
    2. if $Reg_1(a) \in ImList_1$, then abort
    3. $ImList_1 = ImList_1 \cup \{Reg_1(a)\}$
    4. If $(a,b') \in OS$ then set $v = b'$, else sample $b \xleftarrow{\$} \{0,1\}^m$ and set $v = b$. Set $OS = OS \cup \{(a,b)\}$.
    5. if $v - Reg_1(a) \in ImList_2$, then abort
    6. Set $Reg_2(a) = v - Reg_1(a)$.
    7. $ImList_2 = ImList_2 \cup \{Reg_2(a)\}$
  · return $Reg_1(a)$.

* On queries of the form $O_1^{-1}(b)$
  1. if $\nexists x$ st $Reg_1(x) \neq b$.
    (a) $BotList \leftarrow BotList_1 \cup \{b\}$.
    (b) return $\perp$
  2. return $x$ such that $Reg_1(x) = b$.
```

```
* On queries of the form $O_2(a)$
  · if $Reg_2(a)$ is undefined
    1. $Reg_2(a) \xleftarrow{\$} \{0,1\}^m \setminus BotList_2$.
    2. if $Reg_2(a) \in ImList_2$, then abort
    3. $ImList_2 = ImList_2 \cup \{Reg_2(a)\}$
    4. If $(a,b') \in OS$ then set $v = b'$, else sample $b \xleftarrow{\$} \{0,1\}^m$ and set $v = b$. Set $OS = OS \cup \{(a,b)\}$.
    5. if $v - Reg_2(a) \in ImList_1$, then abort
    6. Set $Reg_1(a) = v - Reg_2(a)$.
    7. $ImList_1 = ImList_1 \cup \{Reg_1(a)\}$
  · return $Reg_2(a)$.

* On queries of the form $O_2^{-1}(b)$
  1. if $\nexists x$ st $Reg_2(x) \neq b$.
    (a) $BotList \leftarrow BotList_2 \cup \{b\}$.
    (b) return $\perp$
  2. return $x$ such that $Reg_1(x) = b$.
```

  Since the distribution of outputs in ROra_3 and ROra_4 remains identical,

$$\Pr\left[\mathsf{D}^{\mathsf{H}_4}=1\right]=\Pr\left[\mathsf{D}^{\mathsf{H}_3}=1\right].$$

– **Hybrid** H_5:
- LOra_5 remains identical as LOra_4.
- ROra_5 is identical to ROra_4 except here we abort if a value programmed in $\mathsf{Reg}_2(a)$ during an invocation of a $\mathsf{O}(a)$ already belongs to $\mathsf{BotList}_2$. Formally it is defined as follows:

```
* On queries of the form $O_1(a)$
  · if $Reg_1(a)$ is undefined
    1. $Reg_1(a) \xleftarrow{\$} \{0,1\}^m \backslash BotList_1$.
    2. if $Reg_1(a) \in ImList_1$, then abort
    3. $ImList_1 = ImList_1 \cup \{Reg_1(a)\}$
    4. If $(a, b') \in OS$ then set $v = b'$, else
       sample $b \xleftarrow{\$} \{0,1\}^m$ and set $v = b$. Set
       $OS = OS \cup \{(a, b)\}$.
    5. if $v - Reg_1(a) \in ImList_2$, then abort
    6. Set $Reg_2(a) = v - Reg_1(a)$.
    7. If $Reg_2(a) \in BotList_2$, then abort.
    8. $ImList_2 = ImList_2 \cup \{Reg_2(a)\}$
  · return $Reg_1(a)$.
```

```
* On queries of the form $O_2(a)$
  · if $Reg_2(a)$ is undefined
    1. $Reg_2(a) \xleftarrow{\$} \{0,1\}^m \backslash BotList_2$.
    2. if $Reg_2(a) \in ImList_2$, then abort
    3. $ImList_2 = ImList_2 \cup \{Reg_2(a)\}$
    4. If $(a, b') \in OS$ then set $v = b'$, else
       sample $b \xleftarrow{\$} \{0,1\}^m$ and set $v = b$. Set
       $OS = OS \cup \{(a, b)\}$.
    5. if $v - Reg_2(a) \in ImList_1$, then abort
    6. Set $Reg_1(a) = v - Reg_2(a)$.
    7. If $Reg_1(a) \in BotList_1$, then abort.
    8. $ImList_1 = ImList_1 \cup \{Reg_1(a)\}$
  · return $Reg_2(a)$.
```

```
* On queries of the form $O_1^{-1}(b)$
  1. if $\nexists x$ st $Reg_1(x) \neq b$.
    (a) $BotList \leftarrow BotList \cup \{b\}$.
    (b) return $\perp$
  2. return $x$ such that $Reg_1(x) = b$.
```

```
* On queries of the form $O_2^{-1}(b)$
  1. if $\nexists x$ st $Reg_2(x) \neq b$.
    (a) $BotList \leftarrow BotList \cup \{b\}$.
    (b) return $\perp$
  2. return $x$ such that $Reg_1(x) = b$.
```

Hybrids H_5 and H_4 are identical given the aborts do not occur. Now, each abort statement can be triggered with probability at-most $\frac{q^2}{2^m}$. This is because for the any oracle query, at most q elements can be present in $\mathsf{BotList}_i$ and the probability that for some a, $\mathsf{Reg}_i(a) = b$ for a fixed $b \in \mathsf{BotList}_i$ is $\frac{1}{2^m}$. Taking union bound, we get that at-least one abort statement occurs with probability at-most $\frac{2q^2}{2^m}$. Thus,

$$\left|\Pr\left[\mathsf{D}^{\mathsf{H}_5} = 1\right] - \Pr\left[\mathsf{D}^{\mathsf{H}_4} = 1\right]\right| \leqslant \frac{2q^2}{2^m}.$$

– **Hybrid** H_6:
- LOra_6 remains identical as LOra_5.
- In ROra_6 forward queries are forwarded to oracles $\mathsf{O}_1^*, \mathsf{O}_2^*$.

Here $\mathsf{O}_i^*(a), i \in [2]$ behaves exactly like oracle O_i except it checks the following: Say $b = \mathsf{O}_i(a)$. If there exists any $x \neq a$ such that $\mathsf{O}_i(x) = b$, then it aborts. Note that this condition forces the oracles to behave like an injective random function on the queries received.
The inverse queries are handled identically as ROra_6.
Observe the output distribution of the oracles here is identical to that of the previous hybrid. Therefore,

$$\Pr\left[\mathsf{D}^{\mathsf{H}_6} = 1\right] = \Pr\left[\mathsf{D}^{\mathsf{H}_5} = 1\right].$$

– **Hybrid** H_7:
- LOra_7 remains identical as LOra_6.
- In ROra_6 we replace all simulators and oracle with the real world oracles, i.e., $(\mathsf{O}_1, \mathsf{O}_2, \mathsf{O}_1^{-1}, \mathsf{O}_2^{-1})$.

This differs from the previous hybrid in 2 ways:
- The response to new inverse queries need not be a $\perp$. Now, the probability of a non-bot response by an inverse oracle is upper-bounded by $\frac{1}{2^t}$.

- The abort statement introduced in Hybrid H_6 gets triggered. Let us calculate the probability of this event occurring.
  Let $\mathsf{O}_1(a) = b$. The probability that does not exists any $x \neq a$ such that $\mathsf{O}_1(x) = b$ is exactly

$$\left(1 - \frac{1}{2^{n+t}}\right)^{2^n - 1} \leqslant \left(1 - \frac{1}{2^{n+t}}\right)^{2^{n-1}} \leqslant \frac{c}{e^{\frac{1}{2^{t+1}}}}.$$

  where c is some constant greater than 0. Thus, the probability that the abort conditions do get triggered is at-most $2q \cdot \left(1 - \frac{1}{e^{\frac{c}{2^{t+1}}}}\right)$.

Let $\frac{c}{2^{t+1}}$ be denoted by x, then we get $\left(1 - \frac{1}{e^{\frac{c}{2^{t+1}}}}\right) = \left(1 - \frac{1}{e^x}\right)$, where x becomes smaller as t grows. For $x < 1.79$, we have that $e^x \leqslant 1 + x + x^2$, thus $1 - \frac{1}{e^x} \leqslant 1 - \frac{1}{1+x+x^2} = \frac{x+x^2}{1+x+x^2} \leqslant x + x^2 \leqslant 10x$. x will certainly be less than 1.79 for large enough value of t. This implies $1 - \frac{1}{e^{\frac{c}{2^{t+1}}}} \leqslant \frac{10c}{2^{t+1}}$.
Therefore the total distinguishing probability is

$$\left|\Pr\left[\mathsf{D}^{\mathsf{H}_6} = 1\right] - \Pr\left[\mathsf{D}^{\mathsf{H}_7} = 1\right]\right| \leqslant \frac{2q}{2^t} + \frac{qc}{2^{t+1}} \leqslant \frac{cq}{2^t}.$$

Combining all the hybrids completes the proof.

4.2 Adding Random Permutations

Theorem 4. *Let $\mathsf{O}_1, \mathsf{O}_2$ be random permutations from $\{0,1\}^n$ to $\{0,1\}^n$, $n \in \mathbb{N}$ and $\mathsf{R} : \{0,1\}^n \to \{0,1\}^n$ be a random function. For all polynomial-query-bounded distinguishers D making at most $q = \mathsf{poly}(n)$ queries to each oracle, there exists a poly-time oracle simulator $\mathsf{Sim}^{(\cdot)}$ such that*

$$\left|\Pr\left[\mathsf{D}^{(\mathsf{O}_1+\mathsf{O}_2),(\mathsf{O}_1,\mathsf{O}_2,\mathsf{O}_1^{-1},\mathsf{O}_2^{-1})}(1^n) = 1\right] - \Pr\left[\mathsf{D}^{\mathsf{R},\mathsf{Sim}^{\mathsf{R}}}(1^n) = 1\right]\right| \leqslant \frac{8q^2}{2^n - q} + \frac{2q(2q+1)}{2^n}.$$

Proof. This proof also follows the same hybrid structure as above.

- **Hybrid** H_1: This represents the case where D interacts with oracles $(\mathsf{R}, \mathsf{Sim}^{\mathsf{R}})$.
  - LOra_1 is simply R which on input an n bit binary string outputs a random element from $\{0,1\}^m$.
  - ROra_1 is identical to $\mathsf{Sim}^{\mathsf{R}}$ and is defined as follows:

```
* On queries of the form $\mathsf{O}_1(a)$
    · if $\mathsf{Reg}_1(a)$ is undefined
      1. $\mathsf{Reg}_1(a) \xleftarrow{\$} \{0,1\}^n \backslash \mathsf{ImList}_1$.
      2. $A_1 \leftarrow A_1 \cup \{a\}$.
      3. $\mathsf{ImList}_1 = \mathsf{ImList}_1 \cup \{\mathsf{Reg}_1(a)\}$
      4. Send $a$ as a query to $\mathsf{R}$.
      5. Set $\mathsf{Reg}_2(a) = \mathsf{R}(a) - \mathsf{Reg}_1(a)$.
    · return $\mathsf{Reg}_1(a)$.

* On queries of the form $\mathsf{O}_2(a)$
    · if $\mathsf{Reg}_2(a)$ is undefined
      1. $\mathsf{Reg}_2(a) \xleftarrow{\$} \{0,1\}^n \backslash \mathsf{ImList}_2$.
      2. $A_2 \leftarrow A_2 \cup \{a\}$.
      3. $\mathsf{ImList}_2 = \mathsf{ImList}_2 \cup \{\mathsf{Reg}_2(a)\}$
      4. Send $a$ as a query to $\mathsf{R}$.
      5. Set $\mathsf{Reg}_1(a) = \mathsf{R}(a) - \mathsf{Reg}_2(a)$.
    · return $\mathsf{Reg}_2(a)$.

* On queries of the form $\mathsf{O}_1^{-1}(b)$
    1. if $\nexists x$ st $\mathsf{Reg}_1(x) \neq b$.
      (a) $a \xleftarrow{\$} \{0,1\}^n \backslash A_1$.
      (b) Set $\mathsf{Reg}_1(a) = b$
      (c) $A_1 = A_1 \cup \{a\}$.
      (d) $\mathsf{ImList}_1 = \mathsf{ImList}_1 \cup b$.
      (e) Send $a$ as a query to $\mathsf{R}$.
      (f) Set $\mathsf{Reg}_2(a) = \mathsf{R}(a) - \mathsf{Reg}_1(a)$.
    2. return $a$.

* On queries of the form $\mathsf{O}_2^{-1}(b)$
    1. if $\nexists x$ st $\mathsf{Reg}_2(x) \neq b$.
      (a) $a \xleftarrow{\$} \{0,1\}^n \backslash A_2$.
      (b) Set $\mathsf{Reg}_2(a) = b$
      (c) $A_2 = A_2 \cup \{a\}$.
      (d) $\mathsf{ImList}_2 = \mathsf{ImList}_2 \cup b$.
      (e) Send $a$ as a query to $\mathsf{R}$.
      (f) Set $\mathsf{Reg}_1(a) = \mathsf{R}(a) - \mathsf{Reg}_2(a)$.
    2. return $a$.
```

By definition, we have that

$$\Pr\left[\mathsf{D}^{\mathsf{R},\mathsf{Sim}^{\mathsf{R}}} = 1\right] = \Pr\left[\mathsf{D}^{\mathsf{H}_1} = 1\right]$$

After this, the same hybrid arguments follow as H_2 to H_4 from the proof of Theorem 3.

– **Hybrid** H_5:
  - LOra_5 remains same as LOra_4.
  - ROra_5 is identical to ROra_4 except an additional abort condition collisions occur:

```
* On queries of the form $O_1(a)$
    · if $Reg_1(a)$ is undefined
      1. $b^* \xleftarrow{\$} \{0,1\}^n \backslash ImList_1$.
      2. If $\exists x$ such that $Reg_1(x) = b^*$, then abort else set $Reg_1(a) = b^*$
      3. If $(a, b') \in OS$ then set $v = b'$,
      4. else
         sample $b \xleftarrow{\$} \{0,1\}^n$ and set $v = b$. Set $OS = OS \cup \{(a,b)\}$.
      5. $A_1 \leftarrow A_1 \cup \{a\}$.
      6. $ImList_1 = ImList_1 \cup \{Reg_1(a)\}$
      7. Set $Reg_2(a) = v - Reg_1(a)$.
    · return $Reg_1(a)$.

* On queries of the form $O_2(a)$
    · if $Reg_2(a)$ is undefined
      1. $b^* \xleftarrow{\$} \{0,1\}^n \backslash ImList_2$.
      2. If $\exists x$ such that $Reg_2(x) = b^*$, then abort else set $Reg_2(a) = b^*$
      3. If $(a, b') \in OS$ then set $v = b'$,
      4. else sample $b \xleftarrow{\$} \{0,1\}^n$ and set $v = b$. Set $OS = OS \cup \{(a,b)\}$.
      5. $A_2 \leftarrow A_2 \cup \{a\}$.
      6. $ImList_2 = ImList_2 \cup \{Reg_2(a)\}$
      7. Set $Reg_1(a) = v - Reg_2(a)$.
    · return $Reg_2(a)$.

* On queries of the form $O_1^{-1}(b)$
    1. if $\nexists x$ st $Reg_1(x) \neq b$.
       (a) $a \xleftarrow{\$} \{0,1\}^n \backslash A_1$.
       (b) if $Reg_1(a)$ is defined then abort
       (c) If $(a, b') \in OS$ then set $v = b'$,
       (d) else sample $b \xleftarrow{\$} \{0,1\}^n$ and set $v = b$. Set $OS = OS \cup \{(a,b)\}$.
       (e) Set $Reg_1(a) = b$
       (f) $A_1 = A_1 \cup \{a\}$.
       (g) $ImList_1 = ImList_1 \cup b$.
       (h) Set $Reg_2(a) = R(a) - Reg_1(a)$.
    2. return $a$.

* On queries of the form $O_2^{-1}(b)$
    1. if $\nexists x$ st $Reg_2(x) \neq b$.
       (a) $a \xleftarrow{\$} \{0,1\}^n \backslash A_2$.
       (b) if $Reg_1(a)$ is defined then abort
       (c) If $(a, b') \in OS$ then set $v = b'$,
       (d) else sample $b \xleftarrow{\$} \{0,1\}^n$ and set $v = b$. Set $OS = OS \cup \{(a,b)\}$.
       (e) Set $Reg_2(a) = b$
       (f) $A_2 = A_2 \cup \{a\}$.
       (g) $ImList_2 = ImList_2 \cup b$.
       (h) Set $Reg_1(a) = R(a) - Reg_2(a)$.
    2. return $a$.
```

Conditioned on the fact that the aborts do not occur, hybrids H_3 and H_4 are identical. The probability that the aborts occur is at-most $\frac{8q^2}{2^n-q}$. This is because at any point, the probability that a forward and inverse query will collide is upper bounded by $\frac{2q}{2^n-q}$ and $\frac{2q}{2^n-q}$ respectively. Thus, we get

$$\left|\Pr\left[\mathsf{D}^{\mathsf{H}_5} = 1\right] - \Pr\left[\mathsf{D}^{\mathsf{H}_4} = 1\right]\right| \leqslant \frac{8q^2}{2^n - q}.$$

– **Hybrid** H_6: Let us assume two forward queries a_1, a_2 for O_1 sent by D. $\mathsf{Reg}_2(a_1)$ gets programmed as $OS(a_1) - \mathsf{Reg}_1(a_1)$ and $\mathsf{Reg}_2(a_2) = OS(a_2) - \mathsf{Reg}_1(a_2)$. Although, $OS(a_1) \neq OS(a_2)$ and $\mathsf{Reg}_1(a_1) \neq \mathsf{Reg}_1(a_2)$, there is a non zero probability that $OS(a_1) - \mathsf{Reg}_1(a_1) = OS(a_2) - \mathsf{Reg}_1(a_2)$, which violates the fact that O_2 is a random permutation. Such an inconsistency will also be thrown in during the simulation of the inverses. In order to handle this, we abort whenever such a collision occurs.
  - LOra_6 remains same as LOra_5.
  - ROra_6 is identical to ROra_5 except some additional abort conditions.

```
∗ On queries of the form $\mathsf{O}_1(a)$
  · if $\mathsf{Reg}_1(a)$ is undefined
    1. If $(a, b') \in OS$ then set $v = b'$, else sample $b \xleftarrow{\$} \{0,1\}^n$ and set $v = b$. Set $OS = OS \cup \{(a, b)\}$
    2. $b^* \xleftarrow{\$} \{0,1\}^n \backslash \mathsf{ImList}_1$.
    3. If $\exists x$ such that $\mathsf{Reg}_1(x) = b^*$, then abort else set $\mathsf{Reg}_1(a) = b^*$
    4. $A_1 \leftarrow A_1 \cup \{a\}$.
    5. $\mathsf{ImList}_1 = \mathsf{ImList}_1 \cup \{\mathsf{Reg}_1(a)\}$
    6. if $\exists x$ such that $\mathsf{Reg}_2(x) = OS(a) - \mathsf{Reg}_1(a)$, then abort
    7. Set $\mathsf{Reg}_2(a) = OS(a) - \mathsf{Reg}_1(a)$.
  · return $\mathsf{Reg}_1(a)$.

∗ On queries of the form $\mathsf{O}_2(a)$
  · if $\mathsf{Reg}_2(a)$ is undefined
    1. If $(a, b') \in OS$ then set $v = b'$, else sample $b \xleftarrow{\$} \{0,1\}^n$ and set $v = b$. Set $OS = OS \cup \{(a, b)\}$
    2. $b^* \xleftarrow{\$} \{0,1\}^n \backslash \mathsf{ImList}_2$.
    3. If $\exists x$ such that $\mathsf{Reg}_2(x) = b^*$, then abort else set $\mathsf{Reg}_2(a) = b^*$
    4. $A_2 \leftarrow A_2 \cup \{a\}$.
    5. $\mathsf{ImList}_2 = \mathsf{ImList}_2 \cup \{\mathsf{Reg}_2(a)\}$
    6. if $\exists x$ such that $\mathsf{Reg}_1(x) = OS(a) - \mathsf{Reg}_2(a)$, then abort
    7. Set $\mathsf{Reg}_1(a) = OS(a) - \mathsf{Reg}_2(a)$.
  · return $\mathsf{Reg}_2(a)$.

∗ On queries of the form $\mathsf{O}_1^{-1}(b)$
  1. if $\nexists x$ st $\mathsf{Reg}_1(x) \neq b$.
    (a) $a \xleftarrow{\$} \{0,1\}^n \backslash A_1$.
    (b) if $\mathsf{Reg}_1(a)$ is defined then abort
    (c) If $(a, b') \in OS$ then set $v = b'$, else sample $b \xleftarrow{\$} \{0,1\}^n$ and set $v = b$. Set $OS = OS \cup \{(a, b)\}$
    (d) Set $\mathsf{Reg}_1(a) = b$
    (e) $A_1 = A_1 \cup \{a\}$.
    (f) $\mathsf{ImList}_1 = \mathsf{ImList}_1 \cup b$.
    (g) if $\exists x$ such that $\mathsf{Reg}_2(x) = OS(a) - \mathsf{Reg}_1(a)$, then abort
    (h) Set $\mathsf{Reg}_2(a) = OS(a) - \mathsf{Reg}_1(a)$.
  2. return $a$.

∗ On queries of the form $\mathsf{O}_2^{-1}(b)$
  1. if $\nexists x$ st $\mathsf{Reg}_2(x) \neq b$.
    (a) $a \xleftarrow{\$} \{0,1\}^n \backslash A_2$.
    (b) if $\mathsf{Reg}_1(a)$ is defined then abort
    (c) If $(a, b') \in OS$ then set $v = b'$, else sample $b \xleftarrow{\$} \{0,1\}^n$ and set $v = b$. Set $OS = OS \cup \{(a, b)\}$
    (d) Set $\mathsf{Reg}_2(a) = b$
    (e) $A_2 = A_2 \cup \{a\}$.
    (f) $\mathsf{ImList}_2 = \mathsf{ImList}_2 \cup b$.
    (g) if $\exists x$ such that $\mathsf{Reg}_1(x) = OS(a) - \mathsf{Reg}_2(a)$, then abort
    (h) Set $\mathsf{Reg}_1(a) = OS(a) - \mathsf{Reg}_2(a)$.
  2. return $a$.
```

We argue that the chances that abort occurs is negligible. Since, we are dealing with random permutations, the probability of such collisions is at most $\frac{1}{2^n}$. Since the number of entries in Reg_1 and Reg_2 each are at-most $2q$, the total probability of such collision for all the queries is upper bounded by $\frac{2q(2q+1)}{2^{n+1}}$. This can happen in the simulation of either of the four oracles, hence the probability of bad occurring is at most $\frac{2q(2q+1)}{2^n}$.

$$\left|\Pr\left[\mathsf{D}^{\mathsf{H}_6} = 1\right] \Pr\left[\mathsf{D}^{\mathsf{H}_5} = 1\right]\right| \leqslant \frac{2q(2q+1)}{2^n}.$$

– **Hybrid** H_7:
  - LOra_7 remains same as LOra_6.
  - ROra_7 responds using true oracle $\mathsf{O}_1, \mathsf{O}_1^{-1}, \mathsf{O}_2, \mathsf{O}_2^{-1}$.

  Observe that since there are no collisions and each query is responded with the correct distribution, the output of ROra_6 is indistinguishable from that of true oracle responses. Hence,

$$\Pr\left[\mathsf{D}^{\mathsf{H}_6} = 1\right] = \Pr\left[\mathsf{D}^{\mathsf{H}_7} = 1\right].$$

Combining, we have

$$\left|\Pr\left[\mathsf{D}^{(\mathsf{O}_1+\mathsf{O}_2),(\mathsf{O}_1,\mathsf{O}_2,\mathsf{O}_1^{-1},\mathsf{O}_2^{-1})}(1^n) = 1\right] - \Pr\left[\mathsf{D}^{\mathsf{R},\mathsf{Sim}^{\mathsf{R}}}(1^n) = 1\right]\right| \leqslant \frac{8q^2}{2^n - q} + \frac{2q(2q+1)}{2^n}.$$

5 Constructions from Expanding Oracles with Random Noise

We explore the setup where we add independent random noise to the output of an expanding oracle. Our result concentrates on expanding oracles and we show that our construction achieves indifferentiability from a random function.

Let $\mathsf{O} : \mathbb{F}_{2^n} \to \mathbb{F}_{2^m}^k$ be a random oracle. Analogously define its inverse $\mathsf{O}^{-1} : \mathbb{F}_{2^m}^k \to \mathbb{F}_{2^n} \cup \{\perp\}$. Here $k(n) \in \mathbb{N}$ and $m(n)$ are functions of n. Let $\mathbf{E} = \{e_1, \ldots, e_L\}$ be a publicly known fixed list of length (possibly adversarially generated) L where $e_i \in \mathbb{F}_{2^m}$, $\forall\ i\ \in\ [L]$. Let p be a parameter to a Bernoulli random variable. We define a function $f : \mathbb{F}_{2^n} \to \mathbb{F}_{2^m}^k$ as follows:

1. $y \leftarrow \mathsf{O}(x)$.
2. Let $y = (y_1, y_2, \ldots, y_k), y_i \in \mathbb{F}_{2^m}$.
3. for $i\ \in\ 1 : k$, $b_i \xleftarrow{\$} Ber(p)$.
4. for $i\ \in\ 1 : k$, $\epsilon_i \xleftarrow{\$} \mathbf{E}$, $y'_i \leftarrow (y_i + b_i \cdot \epsilon_i) \mod 2^m$.
5. Merge the new m bit stings as $y' = (y'_1, y'_2, \ldots, y'_k)$.
6. Return y'.

We claim that f is indifferentiable from a random function $\mathsf{R} : \mathbb{F}_{2^n} \to \mathbb{F}_{2^m}^k$.

Theorem 5. *Let $p > 0$ be the parameter of a Bernoulli random variable, $k(n) \in \mathbb{N}, m(n), t(n)$ be functions of n. Let $\mathsf{O} : \mathbb{F}_{2^n} \to \mathbb{F}_{2^m}^k$ and $\mathsf{R} : \mathbb{F}_{2^n} \to \mathbb{F}_{2^m}^k$ be random functions such that $km = n + t$, $kp \geqslant c_1 t$ and $k - kp \geqslant c_2 t$, for some constants $c, c_1, c_2, > 0$. For all polynomial-query-bounded distinguishers $\mathsf{D} := (\mathsf{D}_1, \mathsf{D}_2)$ making at most $q = \mathsf{poly}(n)$ queries to each oracle, there exists a poly-time oracle simulator $\mathsf{Sim}^{(\cdot)}$ and constant $\alpha > 0$, such that*

– *when $L \geqslant 2, p = 1$, we have*

$$\left|\Pr\left[\mathsf{D}_2^{f,(\mathsf{O},\mathsf{O}^{-1})}(1^n, \mathbf{E}) = 1 \middle| \mathbf{E} \leftarrow \mathsf{D}_1(1^n, 1^L)\right] - \Pr\left[\mathsf{D}_2^{\mathsf{R},\mathsf{Sim}^{\mathsf{R}}}(1^n, \mathbf{E}) = 1 \middle| \mathbf{E} \leftarrow \mathsf{D}_1(1^n, 1^L)\right]\right| \leqslant \frac{cq}{2^t} + \frac{q}{L^{c_1 t}},$$

– *and when $L \geqslant 1, 0 < p < 1$, we have*

$$\left|\Pr\left[\mathsf{D}_2^{f,(\mathsf{O},\mathsf{O}^{-1})}(1^n, \mathbf{E}) = 1 \middle| \mathbf{E} \leftarrow \mathsf{D}_1(1^n, 1^L)\right] - \Pr\left[\mathsf{D}_2^{\mathsf{R},\mathsf{Sim}^{\mathsf{R}}}(1^n, \mathbf{E}) = 1 \middle| \mathbf{E} \leftarrow \mathsf{D}_1(1^n, 1^L)\right]\right| \leqslant \frac{(c+1)q}{2^t} + \frac{\alpha q(k+1)}{e^{c_2 t}}.$$

Here f is the function defined above. The above bounds are negligible when $t \in \Omega(\log^{1+\varepsilon} n)$.

Proof. This follows the same structure as the previous indifferentiability proofs in the writeup.

– **Hybrid** H_1: Simulated World.
  - LOra_1 is simply R.
  - ROra_1 is the necessary simulator with oracle access to R that does the following: Let Reg be a data-structure for book-keeping. If $\mathsf{Reg}(a)$ is defined, return $\mathsf{Reg}(a)$. Else, on queries of the form $\mathsf{O}(a)$, $a \in \{0,1\}^n$ forward a to R and store the response as $r \in \mathbb{F}_{2^m}^k$. Record the response as $(r_1, \ldots, r_k)$. Sample n bits $b_1, \ldots, b_k$ such that $b_i \xleftarrow{\$} Ber(p)$. Sample n elements $e_1, \ldots, e_k$ uniformly at random from the fixed error set $\mathbf{E}$. Set $\mathsf{Reg}(a) = (r_1 - b_1 \cdot e_1, \ldots, r_k - b_k \cdot e_k)$. If $\exists x \neq a$ such that $\mathsf{Reg}(x) = (r_1 - b_1 \cdot e_1, \ldots, r_k - b_k \cdot e_k)$, then abort, else Return $\mathsf{Reg}(a)$.
    On queries of the form $\mathsf{O}^{-1}(b)$, $b \in \mathbb{F}_{2^m}^k$, check if $b = \mathsf{Reg}(x)$ for some $x \in \{0,1\}^n$. If yes, then return one of the corresponding pre-images, else return $\perp$.
    By definition of the game, we have,

$$\Pr\left[\mathsf{D}_2^{\mathsf{R},\mathsf{Sim}^{\mathsf{R}}} = 1\right] = \Pr\left[\mathsf{D}_2^{\mathsf{H}_1} = 1\right].$$

– **Hybrid** H_2:

- LOra_2 acts as a forwarding polynomial time oracle Turing Machine which upon input $a \in \{0,1\}^n$ forwards queries to R and outputs its response.
- ROra_2 remains identical to ROra_1.

Clearly,

$$\Pr\left[\mathsf{D}_2^{\mathsf{H}_2} = 1\right] = \Pr\left[\mathsf{D}_2^{\mathsf{H}_1} = 1\right].$$

– **Hybrid** H_3:
  - LOra_3 has oracle access to ROra_3 and upon queried $a \in \mathbb{F}_{2^n}$:
    1. Forward any query a as a forward oracle query to ROra_3, and store the response (call it $y \in \mathbb{F}_{2^m}^k$).
    2. Let $y = (y_1, y_2, \ldots, y_k), y_i \in \mathbb{F}_{2^m}$.
    3. for $i \in 1:k$, $b_i \xleftarrow{\$} Ber(p)$.
    4. for $i \in 1:k$, $\epsilon_i \xleftarrow{\$} \mathbf{E}$, $y_i' \leftarrow (y_i + b_i \cdot \epsilon_i) \mod 2^m$.
    5. $y' = (y_1', y_2' \ldots y_n')$.
    6. Return y'.
  - ROra_3 remains identical as ROra_2.

  Note that the outcomes of LOra_2 and LOra_3 are identically distributed since $\mathsf{R}(a)$ is not revealed and is uniformly distributed.
  Hence,

$$\Pr\left[\mathsf{D}_2^{\mathsf{H}_3} = 1\right] = \Pr\left[\mathsf{D}_2^{\mathsf{H}_2} = 1\right].$$

– **Hybrid** H_4:
  - LOra_4 remains identical as LOra_3.
  - ROra_4 no longer forwards its oracle queries to R. Rather, simply it lazy-samples the responses locally while maintaining consistency in the responses.

  The distribution of outputs in ROra_3 and ROra_4 remains identical.

$$\Pr\left[\mathsf{D}_2^{\mathsf{H}_4} = 1\right] = \Pr\left[\mathsf{D}_2^{\mathsf{H}_3} = 1\right].$$

– **Hybrid** H_5:
  - LOra_5 remains identical as LOra_4.
  - In ROra_5 forward queries are forwarded to oracles $\mathsf{O}_1^*, \mathsf{O}_2^*$. Here $\mathsf{O}_1^*, \mathsf{O}_2^*$ are exactly as defined in Hybrid H_6 in the proof of theorem 3. The inverse queries are handled exactly as ROra_4.

  The distribution of outputs in ROra_4 and ROra_5 remains identical. Thus

$$\Pr\left[\mathsf{D}_2^{\mathsf{H}_5} = 1\right] = \Pr\left[\mathsf{D}_2^{\mathsf{H}_4} = 1\right].$$

– **Hybrid** H_6: Finally, we replace the simulation of inverse and forward queries in ROra_5 with the actual oracles. In other words, H_6 is essentially the case where D has been given access to the oracles $((f^{\mathsf{O}}, (\mathsf{O}, \mathsf{O}^{-1})))$.
  We first take a look at the case when $L \geqslant 2$ and $p = 1$.
  The views of H_5 and H_6 remain identical unless the following events occur: (a) D sends an arbitrary inverse query and receives a non $\perp$ response. (b) D guesses the list of k errors which were sampled to produce the output for f, in which case, D can manufacture a query which would give a non $\perp$ response upon querying for its inverse. (c) O^* never aborts in the previous hybrids. From the proof of theorem 3, we saw that abort happens with probability $\frac{cq}{2^{t+1}}$.
  Now, the probability that an arbitrary inverse query has a non $\perp$ response is at-most $\frac{1}{2^{km-n}}$. The probability that D guesses e_i correctly is upper bounded by $\frac{1}{L^k}$.
  Thus, the upper bound of D's advantage which makes at-most q queries to each oracle is,

$$\frac{q}{2^{km-n}} + \frac{q}{L^k} + \frac{cq}{2^{t+1}} \leqslant \frac{cq}{2^t} + \frac{q}{L^k}$$

Observe that $kp = \Omega(t)$ implies that there exists positive constant c_1 such that $k \geqslant c_1 t$. This gives us the necessary bound.
If $0 < p < 1$, the same three cases above hold. Here, the probability that a random inverse query will receive a non $\perp$ responses remains identical to the previous case. In order to guess the errors e_i correctly, D must guess the b_i correctly for all $i \in [k]$. Assuming that that number of places where $b_i = 1$ is ℓ, the probability of D correctly guessing is $\frac{1}{L^\ell} p^\ell (1-p)^{k-\ell}$. Now, summing over all possible values of ℓ, the total probability of guessing becomes

$$\leqslant \sum_{\ell=0}^{k} \frac{1}{L^\ell} p^\ell (1-p)^{k-\ell}$$
$$\leqslant \sum_{\ell=0}^{k} p^\ell (1-p)^{k-\ell}$$

When $p \leqslant \frac{1}{2}$, the above expression can be written as $(1-p)^k \sum_{\ell=0}^{k} \frac{p^\ell}{(1-p)^\ell}$ which can be upper bounded by $(k+1)(1-p)^k$. Similarly, when $p > 0.5$, the upper bound will be $(k+1)p^k$. Combining these two bounds, we get that the probability of guessing the errors correctly is upper bounded by

$$(k+1) \cdot \max\left((1-p)^k, p^k\right).$$

Since $kp \in \Omega(t)$ and $(k - kp) \in \Omega(t)$, then there exists a positive constant c_2 such that $c_2 t \leqslant \min(kp, k - kp)$, we have that for large enough values of n,

$$(k+1)\max\left((1-p)^k, p^k\right) \leqslant (k+1)\max\left(c' e^{-kp}, c'' e^{-(1-p)k}\right) \leqslant \frac{\alpha(k+1)}{e^{c_2 t}}.$$

, where $c', c'' \geqslant 0$ are constants and $\alpha = \max(c', c'')$. Thus the distinguisher D's advantage is upper bounded by

$$\frac{q}{2^t} + \frac{\alpha q(k+1)}{e^{c_2 t}} + \frac{cq}{2^{t+1}} \leqslant \frac{(c+1)q}{2^t} + \frac{\alpha q(k+1)}{e^{c_2 t}}.$$

Combining these, we get the necessary result.

Remark. Note that if we allow $L = \mathsf{superpoly}(n)$, then k can be any arbitrary integer.

6 Acknowledgements

This research began in a conversation between AS and his brother, Anant Sahai; we are grateful to him for his role in sparking our inquiry into this question. We are grateful to Alexis Korb for pointing out a bug in an earlier version of one of the proofs. We would also like to thank Paul Lou and Yael Eisenberg for several insightful discussions. This research was supported in part from a Simons Investigator Award, DARPA SIEVE award, NTT Research, BSF grant 2012378, a Xerox Faculty Research Award, a Google Faculty Research Award, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through Award HR00112020024.

References

ABDCP15. Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. *Cryptology ePrint Archive*, 2015.
Ajt96. Miklós Ajtai. Generating hard instances of lattice problems. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 99–108, 1996.

Ale03. Michael Alekhnovich. More on average case vs approximation complexity. In *44th Annual Symposium on Foundations of Computer Science*, pages 298–307, Cambridge, MA, USA, October 11–14, 2003. IEEE Computer Society Press.

BBS04. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Crypto*, volume 3152, pages 41–55. Springer, 2004.

BFKL94. Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 278–291, Santa Barbara, CA, USA, August 22–26, 1994. Springer, Heidelberg, Germany.

BGI$^+$01. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.

BM19. Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo random bits. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 227–240. 2019.

BR93. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

BSW11. Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography: 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings 8*, pages 253–273. Springer, 2011.

DH22. Whitfield Diffie and Martin E Hellman. New directions in cryptography. In *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*, pages 365–390. 2022.

DKT16. Dana Dachman-Soled, Jonathan Katz, and Aishwarya Thiruvengadam. 10-round Feistel is indifferentiable from an ideal cipher. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 649–678, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

FY80. Aviezri S. Fraenkel and Yaacov Yesha. Complexity of solving algebraic equations. *Information Processing Letters*, 10(5-Apr):178–179, 1980.

GM19. Shafi Goldwasser and Silvio Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Providing sound foundations for cryptography: on the work of Shafi Goldwasser and Silvio Micali*, pages 173–201. 2019.

GMW87. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 171–185, Santa Barbara, CA, USA, August 1987. Springer, Heidelberg, Germany.

Gol01. Oded Goldreich. *Foundations of Cryptology: Basic Tools*. Cambridge, 2001.

GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206, Victoria, BC, Canada, May 17–20, 2008. ACM Press.

GSW13. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

JLS21. Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 60–73, 2021.

KPG99. Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced oil and vinegar signature schemes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 206–222, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany.

Lyu12. Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 738–755, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

MRH04. Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC 2004: 1st*

Theory of Cryptography Conference, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany.

Pos44. Emil L Post. Recursively enumerable sets of positive integers and their decision problems. 1944.

PS98. Sarvar Patel and Ganapathy S. Sundaram. An efficient discrete log pseudo random generator. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 304–317, Santa Barbara, CA, USA, August 23–27, 1998. Springer, Heidelberg, Germany.

PS19. Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 89–114, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

Rab79. Michael O Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, Massachusetts Inst of Tech Cambridge Lab for Computer Science, 1979.

Reg09. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.

RSA78. Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

Sha49. Claude E Shannon. The synthesis of two-terminal switching circuits. *The Bell System Technical Journal*, 28(1):59–98, 1949.

Sho94. Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science*, pages 124–134, Santa Fe, NM, USA, November 20–22, 1994. IEEE Computer Society Press.

Soa99. Robert I Soare. *Recursively enumerable sets and degrees: A study of computable functions and computably generated sets.* Springer Science & Business Media, 1999.

Tur39. Alan Mathison Turing. Systems of logic based on ordinals. *Proceedings of the London Mathematical Society, Series 2*, 45:161–228, 1939.

Wee20. Hoeteck Wee. Functional encryption for quadratic functions from k-lin, revisited. In *Theory of Cryptography: 18th International Conference, TCC 2020, Durham, NC, USA, November 16–19, 2020, Proceedings, Part I 18*, pages 210–228. Springer, 2020.

Yao82. Andrew C Yao. Protocols for secure computations. In *23rd annual symposium on foundations of computer science (sfcs 1982)*, pages 160–164. IEEE, 1982.