Correcting Subverted Random Oracles

Alexander Russell[1], Qiang Tang[2(✉)], Moti Yung[3], and Hong-Sheng Zhou[4]

[1] University of Connecticut, Mansfield, USA
acr@cse.uconn.edu
[2] New Jersey Institute of Technology, Newark, USA
qiang@njit.edu
[3] Columbia University, New York City, USA
moti@cs.columbia.edu
[4] Virginia Commonwealth University, Richmond, USA
hszhou@vcu.edu

Abstract. The random oracle methodology has proven to be a powerful tool for designing and reasoning about cryptographic schemes, and can often act as an effective bridge between theory and practice. In this paper, we focus on the basic problem of correcting faulty—or adversarially corrupted—random oracles, so that they can be confidently applied for such cryptographic purposes.

We prove that a simple construction can transform a "subverted" random oracle—which disagrees with the original one at a negligible fraction of inputs—into a construction that is *indifferentiable* from a random function. Our results permit future designers of cryptographic primitives in typical kleptographic settings (i.e., with adversaries who may subvert the implementation of cryptographic algorithms but undetectable via blackbox testing) to use random oracles as a trusted black box, in spite of not trusting the implementation. Our analysis relies on a general rejection re-sampling lemma which is a tool of possible independent interest.

1 Introduction

The random oracle methodology [7] has proven to be a powerful tool for designing and reasoning about cryptographic schemes. It consists of the following two steps: (i) design a scheme Π in which all parties (including the adversary) have oracle access to a common truly random function, and establish the security of Π in this favorable setting; (ii) instantiate the random oracle in Π with a suitable cryptographic hash function (such as SHA256) to obtain an instantiated scheme Π'. The random oracle heuristic states that if the original scheme Π is secure, then the instantiated scheme Π' is also secure. While this heuristic can fail in various settings [19] the basic framework remains a fundamental design and analysis tool. In this work we focus on the problem of correcting faulty—or adversarially corrupted—random oracles so that they can be confidently applied for such cryptographic purposes.

H. Shacham and A. Boldyreva (Eds.): CRYPTO 2018, LNCS 10992, pp. 241–271, 2018.
https://doi.org/10.1007/978-3-319-96881-0_9

Specifically, given a function $\tilde{h}$ drawn from a distribution which agrees in most places with a uniform function, we would like to produce a corrected version that has stronger uniformity properties. Our problem shares some features with the classical "self-checking and self-correcting program" paradigm [9–11]: we wish to transform a program that is faulty at a small fraction of inputs (modeling an evasive adversary) to a program that is correct at all points. In this light, our model can be viewed as an adaptation of the classical theory that considers the problem of "self-correcting a probability distribution." Notably, in our setting the functions to be corrected are structureless—specifically, drawn from the uniform distribution—rather than heavily structured. Despite that, the basic procedure for correction and portions of the technical development are analogous.

One particular motivation for correcting random oracles in a cryptographic context arises from recent work studying security in the *kleptographic* setting. In this setting, the various components of a cryptographic scheme may be subverted by an adversary so long as the tampering cannot be detected via black-box testing. This is a challenging setting for a number of reasons highlighted by [6,49,50]: one particular difficulty is that the random oracle paradigm is directly undermined. In terms of the discussion above, the random oracle—which is eventually to be replaced with a concrete function—is subject to adversarial subversion which complicates even the first step (i) of the random oracle methodology above. Our goal is to provide a generic approach that can rigorously "protect" random oracles from subversion.

1.1 Our Contributions

We first give two concrete attacking scenarios where hash functions are subverted in the kleptographic setting. We then express the security properties by adapting the successful framework of *indifferentiability* [23,41] to our setting with adversarial subversion. This framework provides a satisfactory guarantee of modularity—that is, that the resulting object can be directly employed by other constructions demanding a random oracle. We call this new notion "crooked" indifferentiability to reflect the role of adversary in the modeling; see below. (A formal definition appears in Sect. 2.)

We prove that a simple construction involving only *public* randomness can boost a "subverted" random oracle into a construction that is indifferentiable from a random function (Sects. 3 and 4). We remark that our technical development establishes a novel "rejection re-sampling" lemma, controlling the distribution emerging from adversarial re-sampling of product distributions. This may be a technique of independent interest. We expand on these contributions below.

Consequences of kleptographic hash subversion. We first illustrate the damages that are caused by using hash functions that are subverted at only a negligible fraction of inputs with two concrete examples:

(1) Chain take-over attack on blockchain. For simplicity, consider a proof-of-work blockchain setting where miners compete to find a solution s to the "puzzle"

$h(\mathsf{pre}||\mathsf{transactions}||s) \leq d$, where pre denotes the hash of previous block, transactions denotes the set of valid transactions in the current block, and d denotes the difficulty parameter. Here h is intended to be a strong hash function. Note that, the mining machines use a program $\tilde{h}(\cdot)$ (or a dedicated hardware module) which could be designed by a clever adversary. Now if $\tilde{h}$ has been subverted so that $\tilde{h}(*||z) = 0$ for a randomly chosen z—and $\tilde{h}(x) = h(x)$ in all other cases—this will be difficult to detect by prior black-box testing; on the other hand, the adversary who created $\tilde{h}$ has the luxury of solving the proof of work without any effort for any challenge, and thus can completely control the blockchain. (A fancier subversion can tune the "backdoor" z to other parts of the input so that it cannot be reused by other parties; e.g., $\tilde{h}(w||z) = 0$ if $z = f(w)$ for a secret pseudorandom function known to the adversary.)

(2) System sneak-in attack on password authentication. In Unix-style system, during system initialization, the root user chooses a master password α and the system stores the digest $\rho = h(\alpha)$, where h is a given hash function normally modeled as a random oracle. During login, the operating system receives input x and accepts this password if $h(x) = \rho$. An attractive feature of this practice is that it is still secure if ρ is accidentally leaked. In the presence of kleptographic attacks, however, the module that implements the hash function h may be strategically subverted, yielding a new function $\tilde{h}$ which destroys the security of the scheme above: for example, the adversary may choose a relatively short random string z and define $\tilde{h}(y) = h(y)$ *unless* y *begins with* z, in which case $\tilde{h}(zx) = x$. As above, h and $\tilde{h}$ are indistinguishable by black-box testing; on the other hand, the adversary can login as the system administrator using ρ and its knowledge of the backdoor z (without knowing the actual password α, presenting $z\rho$ instead).

The model of "crooked" indifferentiability. The problem of cleaning defective randomness has a long history in computer science. Our setting requires that the transformation must be carried out by a local rule and involve an exponentially small amount of public randomness (in the sense that we wish to clean a defective random function $h : \{0,1\}^n \to \{0,1\}^n$ with only a polynomial length random string). The basic framework of correcting a subverted random oracle is the following:

First, a function $h : \{0,1\}^n \to \{0,1\}^n$ is drawn uniformly at random. Then, an adversary may *subvert* the function h, yielding a new function $\tilde{h}$. The subverted function $\tilde{h}(x)$ is described by an adversarially-chosen (polynomial-time) algorithm $\tilde{H}^h(x)$, with oracle access to h. We insist that $\tilde{h}(x) \neq h(x)$ only at a negligible fraction of inputs.[1] Next, the function $\tilde{h}$ is "publicly corrected" to

[1] We remark that tampering with even a negligible fraction of inputs can have devastating consequences in many settings of interest: e.g., the blockchain and password examples above. Additionally, the setting of negligible subversion is precisely the desired parameter range for existing models of kleptographic subversion and security. In these models, when an oracle is non-negligibly defective, this can be easily detected by a watchdog using a simple sampling and testing regimen, see e.g., [49].

a function $\tilde{h}_R$ (defined below) that involves some public randomness R selected after $\tilde{h}$ is supplied.[2]

We wish to show that the resulting function (construction) is "as good as" a random oracle, in the sense of indifferentiability. We say a construction C^H (having oracle access to an ideal primitive H) is indifferentiable from another ideal primitive $\mathcal{F}$, if there exists a simulator $\mathcal{S}$ so that (C^H, H) and $(\mathcal{F}, \mathcal{S})$ are indistinguishable to any distinguisher $\mathcal{D}$.

To reflect our setting, an H-crooked-distinguisher $\widehat{\mathcal{D}}$ is introduced; the H-crooked-distinguisher $\widehat{\mathcal{D}}$ first prepares the subverted implementation $\tilde{H}$ (after querying H first); then a fixed amount of (public) randomness R is drawn and published; the construction C uses only subverted implementation $\tilde{H}$ and R. Now following the indifferentiability framework, we will ask for a simulator $\mathcal{S}$, such that $(C^{\tilde{H}^H}(\cdot, R), H)$ and $(\mathcal{F}, \mathcal{S}^{\tilde{H}}(R))$ are indistinguishable to any H-crooked-distinguisher $\widehat{\mathcal{D}}$ who even knows R. A similar security preserving theorem [23,41] also holds in our model. See Sect. 2 for details.

The construction. The construction depends on a parameter $\ell = \text{poly}(n)$ and public randomness $R = (r_1, \ldots, r_\ell)$, where each r_i is an independent and uniform element of $\{0,1\}^n$. For simplicity, the construction relies on a family of independent random oracles $h_i(x)$, for $i \in \{0, \ldots, \ell\}$. (Of course, these can all be extracted from a single random oracle with slightly longer inputs by defining $\tilde{h}_i(x) = \tilde{h}(i, x)$ and treating the output of $h_i(x)$ as n bits long.) Then we define

$$\tilde{h}_R(x) = \tilde{h}_0\left(\bigoplus_{i=1}^{\ell} \tilde{h}_i(x \oplus r_i)\right) = \tilde{h}_0\Big(\tilde{g}_R(x)\Big).$$

Note that the adversary is permitted to subvert the function(s) h_i by choosing an algorithm $H^{h_*}(x)$ so that $\tilde{h}_i(x) = H^{h_*}(i, x)$. Before diving into the analysis, let us first quick demonstrate how some simpler constructions fail.

Simple constructions and their shortcomings. Although during the stage of manufacturing the hash functions $\tilde{h}_* = \{\tilde{h}_i\}_{i=0}^{\ell}$, the randomness $R := r_1, \ldots, r_\ell$ are not known to the adversary, they become public in the second query phase. If the "mixing" operation is not carefully designed, the adversary could choose inputs accordingly, trying to "peel off" R. We discuss a few examples:

1. $\tilde{h}_R(x)$ is simply defined as $\tilde{h}_1(x \oplus r_1)$. A straightforward attack is as follows: the adversary can subvert h_1 in a way that $\tilde{h}_1(m) = 0$ for a random input m; the adversary then queries $m \oplus r_1$ on $\tilde{h}_R(\cdot)$ and can trivially distinguish $\tilde{h}$ from a random function.

[2] We remark that in many settings, e.g., the model of classical self-correcting programs, we are permitted to sample fresh and "private" randomness for each query; in our case, we may only use a single polynomial-length random string for all points. Once R is generated, it is made public and fixed, which implicitly defines our corrected function $\tilde{h}_R(\cdot)$. This latter requirement is necessary in our setting as random oracles are typically used as a public object—in particular, our attacker must have full knowledge of R.

2. $\tilde{h}_R(x)$ is defined as $\tilde{h}_1(x \oplus r_1) \oplus \tilde{h}_2(x \oplus r_2)$. Now a slightly more complex attack can still succeed: the adversary subverts h_1 so that $\tilde{h}_1(x) = 0$ if $x = m||*$, that is, when the first half of x equals to a randomly selected string m with length $n/2$; likewise, h_2 is subverted so that $\tilde{h}_2(x) = 0$ if $x = *||m$, that is, the second half of x equals m. Then, the adversary queries $m_1||m_2$ on $\tilde{h}_R(\cdot)$, where $m_1 = m \oplus r_{1,0}$, and $m_2 = m \oplus r_{2,1}$, and $r_{1,0}$ is the first half of r_1, and $r_{2,1}$ is the second half of r_2. Again, trivially, it can be distinguished from a random function.
   This attack can be generalized in a straightforward fashion to any $\ell \leq n/\lambda$: the input can be divided in into consecutive substrings each with length λ, and the "trigger" substrings can be planted in each chunk.

Challenges in the analysis. To analyze security in the "crooked" indifferentiability framework, our simulator needs to ensure consistency between two ways of generating output values: one is directly from the construction $C^{\tilde{H}^h}(x, R)$; the other calls for an "explanation" of F—a truly random function—via reconstruction from related queries to H (in a way consistent with the subverted implementation $\tilde{H}$). To ensure a correct simulation, the simulator must suitably answer related queries (defining one value of $C^{\tilde{H}^h}(x, R)$). We develop a theorem establishing an unpredictability property of the internal function $\tilde{g}_R(x)$ to guarantee the success of simulation. In particular, we prove that for any input x (if not yet "fully decided" by previous queries), the output of $\tilde{g}_R(x)$ is unpredicatable to the distinguisher even if she knows the public randomness R (even conditioned on adaptive queries generated by $\widehat{\mathcal{D}}$).

Section 4 develops the detailed security analysis for the property of the internal function $\tilde{g}_R(x)$. The proof of correctness for this construction is complicated by the fact that the "defining" algorithm $\tilde{H}$ is permitted to make adaptive queries to h during the definition of $\tilde{h}$; in particular, this means that even when a particular "constellation" of points (the $h_i(x \oplus r_i)$) contains a point that is left alone by $\tilde{H}$ (which is to say that it agrees with $h_i()$) there is no guarantee that $\bigoplus_i h_i(x \oplus r_i)$ is uniformly random. This suggests focusing the analysis on demonstrating that the constellation associated with every $x \in \{0,1\}^n$ will have at least one "good" component, which is (i.) not queried by $\tilde{H}^h(\cdot)$ when evaluated on the other terms, and (ii.) answered honestly. Unfortunately, actually identifying such a good point with certainty appears to require that we examine *all* of the points in the constellation for x, and this interferes with the standard "exposure martingale" proof that is so powerful in the random oracle setting (which capitalizes on the fact that "unexamined" values of h can be treated as independent and uniform values).

To sidestep this difficulty, we prove a "resampling" lemma, which lets us examine all points in a particular constellation, identify one "good" one of interest, and then *resample* this point so as to "forget" about all possible conditioning this value might have. The resampling lemma gives a precise bound on the effects of such conditioning.

Immediate applications: Our correction function can be easily applied to save the faulty hash implementation in several important application scenarios, as explained in the motivational examples.

(1) For proof-of-work based blockchains, as discussed above, miners may rely on a common library $\tilde{h}$ for the hash evaluation, perhaps cleverly implemented by an adversary. Here $\tilde{h}$ is determined before the chain has been deployed. We can then prevent the adversary from capitalizing on this subversion by applying our correction function. In particular, the public randomness R can be embedded in the genesis block; the function $\tilde{h}_R(\cdot)$ is then used for mining (and verification) rather than $\tilde{h}$.
(2) The system sneak-in can also be resolved immediately by applying our correcting random oracle. During system initialization (or even when the operating system is released), the system administrator generates some randomness R and wraps the hash module $\tilde{h}$ (potentially subverted) to define $\tilde{h}_R(\cdot)$. The password α then gives rise to the digest $\rho = \tilde{h}_R(\alpha)$ together with the randomness R. Upon receiving input x, the system first "recovers" $\tilde{h}_R(\cdot)$ based on the previously stored R, and then tests if $\rho = \tilde{h}_R(x)$. The access will be enabled if the test is valid. As the corrected random oracle ensures the output to be uniform for every input point, this remains secure in the face of subversion.[3]

1.2 Related Work

Related work on indifferentiability. The notion of indifferentiability was proposed by Maurer et al. [41], as an extension of the classical concept of indistinguishability when one or more oracles are publicly available (such as a random oracle). It was later adapted by Coron et al. [23] and generalized to several other variants in [29,33,34,47,53]. Notably, a line of elegant work demonstrated the equivalence of the random oracle model and the ideal cipher model; in particular, the Feistel construction (with a small constant number of rounds) is indifferentiable from an ideal cipher, see [24–26]. Our work adapts the indifferentiability framework to the setting where the construction uses only a subverted implementation, which we call "crooked indifferentiability," where the construction aims to be indifferentiable from another repaired random oracle.

Related work on self-correcting programs. The theory of program self-testing, and self-correcting, was pioneered by the work of Blum et al. [9–11]. This theory addresses the basic problem of program correctness by verifying relationships between the outputs of the program on randomly selected inputs; a similar problem is to turn an almost correct program into one that is correct at every point with overwhelming probability. Rubinfeld's thesis [48] is an authoritative survey of the basic framework and results. Our results can be seen as a distributional analogue of this theory but with two main differences: (i). we insist on using

[3] Typical authentication of this form also uses password "salt," but this doesn't change the structure of the attack or the solution.

only public randomness drawn once for the entire "correction"; (ii). our target object is a distribution, rather than a particular function.

Related work on random oracles. The random oracle methodology [7] can significantly simplify both cryptographic constructions and proofs, even though there exist schemes which are secure using random oracles, but cannot be instantiated in the standard model, [19]. On the other hand, efforts have been made to identify instantiable assumptions/models in which we may analyze interesting cryptographic tasks [4,12–14,16,18,20,39]. Also, we note that research efforts have also been made to investigate *weakened* idealized models [37,38,40,45]. Finally, there are several recent nice works about random oracle in the auxiliary input model (or with pre-processing) [31,51]. Our model shares some similarities that the adversary may embed some preprocessed information into the subverted implementation, but our subverted implementation can further misbehave. Our results strengthen the random oracle methodology in the sense that using our construction, we can even tolerate a faulty hash implementation.

Related work on kleptographic security. Kleptographic attacks were originally introduced by Young and Yung [54,55]; In such attacks, the adversary provides subverted implementations of the cryptographic primitive, trying to learn secret without being detected. In recent years, several remarkable allegations of cryptographic tampering [42,46], including detailed investigations [21,22], have produced a renewed interest in both kleptographic attacks and in techniques for preventing them [1–3,5,6,8,15,27,28,30,32,43,49,50,52]. None of those work considered how to actually correct a subverted random oracle.

Concurrently, Fischlin et al. [36] also considered backdoored (keyed) hash functions, and how to immunize them particularly for the settings of HMAC and HKDF. They focused on preserving some weaker property of weak pseudorandomness for the building block of the compression function. We aim at correcting all the properties of a subverted random oracle, and moreover, our correction function can be applied to immunize backdoored *public* hash functions, which was left open in [36].

Similar constructions in other context. Our construction follows the simple intuition by mixing and input and output by XORing multiple terms. This share similarities in constructions in several other scenarios, e.g., about hardness amplification, notably the famous Yao XOR lemma, and for weak PRF [44]; and randomizers in the bounded storage model [35]. Our construction has to have an external layer of h_0 to wrap the XOR of terms, and our analysis is very different from them due to that our starting point of a subverted implementation.

2 The Model: Crooked Indifferentiability

2.1 Preliminary: Indifferentiability

The notion of indifferentiability introduced by Maurer et al. [41] has been found very useful for studying the security of hash function and many other primitives, especially model them as idealized objectives. This notion is an extension of the

classical notion of indistinguishability, when one or more oracles are publicly available. The indifferentiability notion in [41] is given in the framework of random systems providing interfaces to other systems. Coron et al. [23] demonstrate an equivalent indifferentiability notion for random oracles but in the framework of Interactive Turing Machines (as in [17]). The indifferentiability formulation in this subsection is essentially taken from [23]. In the next subsection, we will introduce our new notion, *crooked indifferentiability.*

Defining indifferentiability. We consider ideal primitives. An ideal primitive is an algorithmic entity which receives inputs from one of the parties and returns its output immediately to the querying party. We now proceed to the definition of indifferentiability [23,41]:

Definition 1 (Indifferentiability [23,41]). *A Turing machine C with oracle access to an ideal primitive $\mathcal{G}$ is said to be $(t_{\mathcal{D}}, t_{\mathcal{S}}, q, \epsilon)$-indifferentiable from an ideal primitive $\mathcal{F}$, if there is a simulator $\mathcal{S}$, such that for any distinguisher $\mathcal{D}$, it holds that :*

$$\left|\Pr[\mathcal{D}^{C,\mathcal{G}} = 1] - \Pr[\mathcal{D}^{\mathcal{F},\mathcal{S}} = 1]\right| \leq \epsilon\,.$$

The simulator $\mathcal{S}$ has oracle access to $\mathcal{F}$ and runs in time at most $t_{\mathcal{S}}$. The distinguisher $\mathcal{D}$ runs in time at most $t_{\mathcal{D}}$ and makes at most q queries. Similarly, $C^{\mathcal{G}}$ is said to be (computationally) indifferentiable from $\mathcal{F}$ if ϵ is a negligible function of the security parameter λ (for polynomially bounded $t_{\mathcal{D}}$ and $t_{\mathcal{S}}$). See Fig. 1.

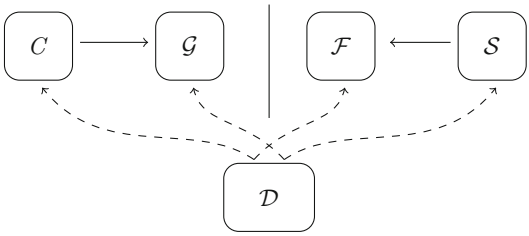

Fig. 1. The indifferentiability notion: the distinguisher $\mathcal{D}$ either interacts with algorithm C and ideal primitive $\mathcal{G}$, or with ideal primitive $\mathcal{F}$ and simulator $\mathcal{S}$. Algorithm C has oracle access to $\mathcal{G}$, while simulator $\mathcal{S}$ has oracle access to $\mathcal{F}$.

As illustrated in Fig. 1, the role of the simulator is to simulate the ideal primitive $\mathcal{G}$ so that no distinguisher can tell whether it is interacting with C and $\mathcal{G}$, or with $\mathcal{F}$ and $\mathcal{S}$; in other words, the output of $\mathcal{S}$ should look "consistent" with what the distinguisher can obtain from $\mathcal{F}$. Note that normally the simulator does not see the distinguisher's queries to $\mathcal{F}$; however, it can call $\mathcal{F}$ directly when needed for the simulation.

Replacement. It is shown in [41] that if $C^{\mathcal{G}}$ is indifferentiable from $\mathcal{F}$, then $C^{\mathcal{G}}$ can replace $\mathcal{F}$ in any cryptosystem, and the resulting cryptosystem is at least *as secure* in the $\mathcal{G}$ model as in the $\mathcal{F}$ model.

We use the definition of [41] to specify what it means for a cryptosystem to be at least as secure in the $\mathcal{G}$ model as in the $\mathcal{F}$ model. A cryptosystem is modeled as an Interactive Turing Machine with an interface to an adversary $\mathcal{A}$ and to a public oracle. The cryptosystem is run by an environment $\mathcal{E}$ which provides a binary output and also runs the adversary. In the $\mathcal{G}$ model, cryptosystem $\mathcal{P}$ has oracle access to C (which has oracle access to $\mathcal{G}$), whereas attacker $\mathcal{A}$ has oracle access to $\mathcal{G}$. In the $\mathcal{F}$ model, both $\mathcal{P}$ and $\mathcal{S}_{\mathcal{A}}$ (the simulator) has direct oracle access to $\mathcal{F}$. The definition is illustrated in Fig. 2.

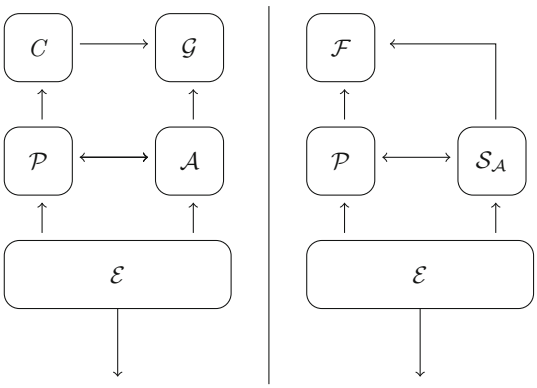

Fig. 2. The environment $\mathcal{E}$ interacts with cryptosystem $\mathcal{P}$ and attacker $\mathcal{A}$. In the $\mathcal{G}$ model (left), $\mathcal{P}$ has oracle access to C whereas $\mathcal{A}$ has oracle access to $\mathcal{G}$. In the $\mathcal{F}$ model, both $\mathcal{P}$ and $\mathcal{S}_{\mathcal{A}}$ have oracle access to $\mathcal{F}$.

Definition 2. *A cryptosystem $\mathcal{P}$ is said to be at least as secure in the $\mathcal{G}$ model with algorithm C, as in the $\mathcal{F}$ model, if for any environment $\mathcal{E}$ and any attacker $\mathcal{A}$ in the $\mathcal{G}$ model, there exists an attacker $\mathcal{S}_{\mathcal{A}}$ in the $\mathcal{F}$ model, such that:*

$$\Pr[\mathcal{E}(\mathcal{P}^{C}, \mathcal{A}^{\mathcal{G}}) = 1] - \Pr[\mathcal{E}(\mathcal{P}^{\mathcal{F}}, \mathcal{S}_{\mathcal{A}}^{\mathcal{F}}) = 1] \leq \epsilon.$$

where ϵ is a negligible function of the security parameter λ, and the notation $\mathcal{E}(\mathcal{P}^{C}, \mathcal{A}^{\mathcal{G}})$ defines the output of $\mathcal{E}$ after interacting with $\mathcal{P}, \mathcal{A}$ as on the left side of Fig. 2 (similarly we can define the right hand side). Moreover, a cryptosystem is said to be computationally at least as secure, etc., if $\mathcal{E}$, $\mathcal{A}$ and $\mathcal{S}_{\mathcal{A}}$ are polynomial-time in λ.

We have the following security preserving (replacement) theorem, which says that when an ideal primitive is replaced by an indifferentiable one, the security of the "bigger" cryptosystem remains:

Theorem 1 ([23,41]). *Let $\mathcal{P}$ be a cryptosystem with oracle access to an ideal primitive $\mathcal{F}$. Let C be an algorithm such that $C^{\mathcal{G}}$ is indifferentiable from $\mathcal{F}$. Then cryptosystem $\mathcal{P}$ is at least as secure in the $\mathcal{G}$ model with algorithm C as in the $\mathcal{F}$ model.*

2.2 Crooked Indifferentiability

The ideal primitives that we focus on in this paper are random oracles. A random oracle [7] is an ideal primitive which provides a random output for each new query, and for the identical input queries the same answer will be given. Next we will formalize a new notion called crooked indifferentiability to characterize subversion. For simplicity, our formalization here is for random oracles, we remark that the formalization can be easily extended for other ideal primitives.

Crooked indifferentiability for random oracles. Let us briefly recall our goal: as mentioned in the Introduction, we are considering to repair a subverted/faulty random oracle, such that the corrected construction can be used as good as a random oracle. It is thus natural to consider the indifferentiability notion. However, we need to adjust the notion to properly model the subversion and to avoid trivial impossibility.

We use H to denote the original random oracle and $\tilde{H}_z$ to be the subverted implementation (where z could be the potential backdoor hardcoded in the implementation and we often ignore it using $\tilde{H}$ for simplicity). There will be several modifications to the original indifferentiability notion. (1) The deterministic construction C will have the oracle access to the random oracle via the subverted implementation $\tilde{H}$, not via the original ideal primitive H; This creates lots of difficulty (and even impossibility) for us to develop a suitable construction. For that reason, the construction is allowed to access to trusted but public randomness r (see Remark 1 below). (2) The simulator will also have the oracle access to the subverted implementation $\tilde{H}$ and also the public randomness r. Item (2) is necessary as it is clearly impossible to have an indifferentiability definition with a simulator that has no access to $\tilde{H}$, as the distinguisher can simply make query an input such that C will use a value that is modified by $\tilde{H}$ while $\mathcal{S}$ has no way to reproduce it. More importantly, we will show below that, the security will still be preserved to replace an ideal random oracle with a construction satisfying our definition (with an augmented simulator). We will prove the security preserving (i.e., replacement) theorem from [23,41] similarly with our adapted notions. (3) To model the whole process of the subversion and correction, we consider a two-stage adversary: subverting and distinguishing. For simplicity, we simply consider them as parts of one distinguisher, and do not use separate the notations and state passing.

Definition 3 (H-crooked indifferentiability). *Consider a distinguisher $\widehat{\mathcal{D}}$ and the following multi-phase real execution. Initially, the distinguisher $\widehat{\mathcal{D}}$ who has oracle access to ideal primitive H, publishes a subverted implementation of H (denoted as $\tilde{H}$). Secondly, a uniformly random string r is sampled and published.*

Thirdly, a deterministic construction C is then developed: the construction C has random string r as input, and has the oracle access to $\tilde{H}$ (the crooked version of H). Finally, the distinguisher $\widehat{\mathcal{D}}$, also having random string r as input, and the oracle access to the pair (C, H), returns a decision bit b. Often, we call $\widehat{\mathcal{D}}$, the H-crooked-distinguisher.

In addition, consider the corresponding multi-phase ideal execution with the same H-crooked-distinguisher $\widehat{\mathcal{D}}$. In the ideal execution, ideal primitive $\mathcal{F}$ is provided. The first two phases are the same (as that in the real execution). In the third phase, a simulator $\mathcal{S}$ will be developed: the simulator has a random string r as input, and has the oracle access to $\tilde{H}$, as well as the ideal primitive $\mathcal{F}$. In the last phase, the H-crooked-distinguisher $\widehat{\mathcal{D}}$, after having random string r as input, and having the oracle access to an alternative pair $(\mathcal{F}, \mathcal{S})$, returns a decision bit b.

We say that construction C, is $(t_{\widehat{\mathcal{D}}}, t_{\mathcal{S}}, q, \epsilon)$-$H$-crooked-indifferentiable from ideal primitive $\mathcal{F}$, if there is a simulator $\mathcal{S}$ so that for any H-crooked-distinguisher $\widehat{\mathcal{D}}$, (let u be the coins of $\widehat{\mathcal{D}}$), it satisfies that the real execution and the ideal execution are indistinguishable. Specifically,

$$\left| \Pr_{u,r,H}\left[\tilde{H} \leftarrow \widehat{\mathcal{D}} : \widehat{\mathcal{D}}^{C^{\tilde{H}}(r),H}(\lambda, r) = 1\right] - \Pr_{u,r,\mathcal{F}}\left[\tilde{H} \leftarrow \widehat{\mathcal{D}} : \widehat{\mathcal{D}}^{\mathcal{F},\mathcal{S}^{\tilde{H},\mathcal{F}}(r)}(\lambda, r) = 1\right]\right| \leq \epsilon(\lambda)\,.$$

Here $H : \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$ and $\mathcal{F} : \{0,1\}^{k} \to \{0,1\}^{k}$ denote random functions. See Fig. 3 for detailed illustration of the last phase in both real and ideal executions (the distinguishing).

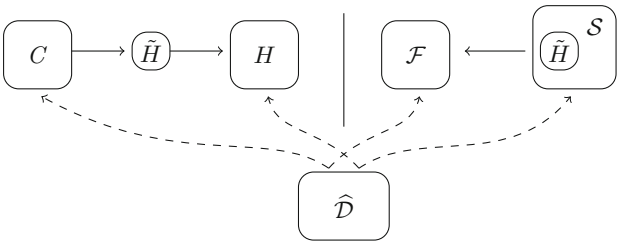

Fig. 3. The H-crooked indifferentiability notion: the distinguisher $\widehat{\mathcal{D}}$, in the first phase, manufactures and publishes a subverted implementation denoted as $\tilde{H}$, for ideal primitive H; then in the second phase, a random string r is published; after that, in the third phase, construction C, or simulator $\mathcal{S}$ is developed; the H-crooked-distinguisher $\widehat{\mathcal{D}}$, in the last phase, either interacting with algorithm C and ideal primitive H, or with ideal primitive $\mathcal{F}$ and simulator $\mathcal{S}$, return a decision bit. Here, algorithm C has oracle access to $\tilde{H}$, while simulator $\mathcal{S}$ has oracle access to $\mathcal{F}$ and $\tilde{H}$.

Remark 1 (The necessity of public randomness). It appears difficult to achieve feasibility without randomness. Intuitively: suppose the corrected hash is represented as $g\big(\tilde{H}(f(x))\big)$, i.e., $g(\cdot), f(\cdot)$ are correction functions, f will be applied to each input x before calling $\tilde{H}$, and g will be further applied to the corresponding output; then the attacker can plant a trigger using $f(z)$ for a random backdoor z, such that $\tilde{H}\left(f(z)\right) = 0$. It is easy to see that the attacker who has full knowledge of $(z, g(0))$ and can use this pair to distinguish, as $\mathcal{F}(z)$ would be a random value that would not hit $g(0)$ with a noticeable probability. Similarly, we can see that it is also infeasible if the randomness is generated before the faulty implementation is provided. For this reason, we allow the model to have a public randomness that is generated after $\tilde{H}$ is supplied, but such randomness would be available to everyone, including the attacker.

Remark 2 (Comparison with preprocessing). There have been several recent nice works [31,51] about random oracle with preprocessing, in which the adversary can have some auxiliary input compressing the queries. While in the first phase of our model, we also allow the adversary to generate such an auxiliary string as part of the backdoor z (or part of the instruction of $\tilde{H}$). We further allow the crooked implementation to deviate from the original random oracle. In this sense, the preprocessing model for random oracle can be considered to defend against a similar attacker than us, but the attacker would provide an honest implementation (only treating the backdoor as the auxiliary input). We note that their construction using simple salting mechanism [51] cannot correct a subverted random oracle as in our model: the distinguisher plants a trigger z into the inputs that $\tilde{H}(z||*) = 0$ for a randomly chosen z. In this way, the salt would be subsumed into the $*$ part and has no effect on the faulty implementation.

Remark 3 (Extensions). For simplicity, our definition is mainly for random oracle. It is not very difficult to extend our crooked indifferentiability notion to the other setting such as ideal cipher, as long as we represent the interfaces properly, while the multi-phase executions can be similarly defined. Another interesting extension is to consider a global random oracle (while in the current definition, there would be an independent instance in the real and ideal execution). We leave those interesting questions to be explored in future works.

Replacement with crooked indifferentiability. Security preserving (replacement) has been shown in the indifferentiability framework [41]: if $C^{\mathcal{G}}$ is indifferentiable from $\mathcal{F}$, then $C^{\mathcal{G}}$ can replace $\mathcal{F}$ in any cryptosystem, and the resulting cryptosystem in the $\mathcal{G}$ model is at least as secure as that in the $\mathcal{F}$ model. We next show that the replacement property can also hold in our crooked indifferentiability framework. Recall that, in the "standard" indifferentiability framework [23,41], a cryptosystem can be modeled as an Interactive Turing Machine with an interface to an adversary $\mathcal{A}$ and to a public oracle. There the cryptosystem is run by a "standard" environment $\mathcal{E}$ (see Fig. 2). In our crooked indifferentiability framework, a cryptosystem also has the interface to an adversary $\mathcal{A}$ and to a public oracle. However, now the cryptosystem is run by a environment $\widehat{\mathcal{E}}$ that can "crook" the oracle.

Consider an ideal primitive $\mathcal{G}$. Similar to the $\mathcal{G}$-crooked-distinguisher, we can define the $\mathcal{G}$-crooked-environment $\widehat{\mathcal{E}}$ as follows: Initially, the $\mathcal{G}$-crooked-environment $\widehat{\mathcal{E}}$ manufactures and then publishes a subverted implementation of the ideal primitive $\mathcal{G}$, denoted as $\tilde{\mathcal{G}}$. Then $\widehat{\mathcal{E}}$ runs the attacker $\mathcal{A}$, and the cryptosystem $\mathcal{P}$ is developed. In the $\mathcal{G}$ model, cryptosystem $\mathcal{P}$ has oracle access to C whereas attacker $\mathcal{A}$ has oracle access to $\mathcal{G}$; note that, C has oracle access to $\tilde{\mathcal{G}}$, not directly to $\mathcal{G}$. In the $\mathcal{F}$ model, both $\mathcal{P}$ and $\mathcal{S}_{\mathcal{A}}$ (the simulator) have oracle access to $\mathcal{F}$. Finally, the $\mathcal{G}$-crooked-environment $\widehat{\mathcal{E}}$ returns a binary decision output. The definition is illustrated in Fig. 4.

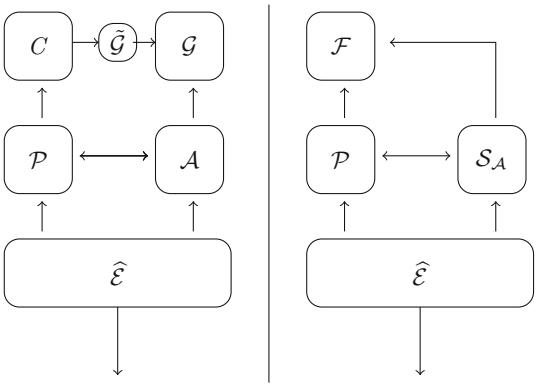

Fig. 4. The environment $\widehat{\mathcal{E}}$ interacts with cryptosystem $\mathcal{P}$ and attacker $\mathcal{A}$. In the $\mathcal{G}$ model (left), $\mathcal{P}$ has oracle access to C (who has oracle access to $\mathcal{G}$) whereas $\mathcal{A}$ has oracle access to $\mathcal{G}$; In the $\mathcal{F}$ model, both $\mathcal{P}$ and $\mathcal{S}_{\mathcal{A}}$ have oracle access to $\mathcal{F}$.

Definition 4. *Consider ideal primitives $\mathcal{G}$ and $\mathcal{F}$. A cryptosystem $\mathcal{P}$ is said to be at least as secure in the $\mathcal{G}$-crooked model with algorithm C as in the $\mathcal{F}$ model, if for any $\mathcal{G}$-crooked-environment $\widehat{\mathcal{E}}$ and any attacker $\mathcal{A}$ in the $\mathcal{G}$-crooked model, there exists an attacker $\mathcal{S}_{\mathcal{A}}$ in the $\mathcal{F}$ model, such that:*

$$\Pr[\widehat{\mathcal{E}}(\mathcal{P}^{C^{\tilde{\mathcal{G}}}}, \mathcal{A}^{\mathcal{G}}) = 1] - \Pr[\widehat{\mathcal{E}}(\mathcal{P}^{\mathcal{F}}, \mathcal{S}_{\mathcal{A}}^{\mathcal{F}}) = 1] \leq \epsilon.$$

where ϵ is a negligible function of the security parameter λ, and $\widehat{\mathcal{E}}(\mathcal{P}^{C^{\tilde{\mathcal{G}}}}, \mathcal{A}^{\mathcal{G}})$ describes the output of $\widehat{\mathcal{E}}$ running the experiment in the $\mathcal{G}$-world (the left side of Fig. 4), and similarly for $\widehat{\mathcal{E}}(\mathcal{P}^{\mathcal{F}}, \mathcal{S}_{\mathcal{A}}^{\mathcal{F}})$.

We now demonstrate the following theorem which shows that security is preserved when replacing an ideal primitive by a crooked-indifferentiable one:

Theorem 2. *Let $\mathcal{P}$ be a cryptosystem with oracle access to an ideal primitive $\mathcal{F}$. Let C be an algorithm such that $C^{\mathcal{G}}$ is crooked-indifferentiable from $\mathcal{F}$. Then cryptosystem $\mathcal{P}$ is at least as secure in the $\mathcal{G}$-crooked model with algorithm C as in the $\mathcal{F}$ model.*

Proof. The proof is very similar to that in [23,41]. Let $\mathcal{P}$ be any cryptosystem, modeled as an Interactive Turing Machine. Let $\widehat{\mathcal{E}}$ be any crooked-environment, and $\mathcal{A}$ be any attacker in the $\mathcal{G}$-crooked model. In the $\mathcal{G}$-crooked model, $\mathcal{P}$ has oracle access to C (who has oracle access to $\tilde{\mathcal{G}}$, not directly to $\mathcal{G}$.), whereas $\mathcal{A}$ has oracle access to ideal primitive $\mathcal{G}$; moreover, the crooked-environment $\widehat{\mathcal{E}}$ interacts with both $\mathcal{P}$ and $\mathcal{A}$. This is illustrated in Fig. 5 (left part).

Since C is crooked-indifferentiable from $\mathcal{F}$ (see Fig. 3), one can replace $(C^{\tilde{\mathcal{G}}}, \mathcal{G})$ by $(\mathcal{F}, \mathcal{S})$ with only a negligible modification of the $\mathcal{G}$-crooked-environment $\widehat{\mathcal{E}}$'s output distribution. As illustrated in Fig. 5, by merging attacker $\mathcal{A}$ and simulator $\mathcal{S}$, one obtains an attacker $\mathcal{S}_{\mathcal{A}}$ in the $\mathcal{F}$ model, and the difference in $\widehat{\mathcal{E}}$'s output distribution is negligible. □

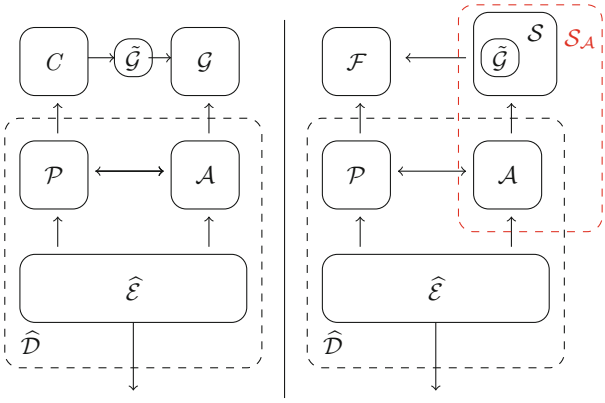

Fig. 5. Construction of attacker $\mathcal{S}_{\mathcal{A}}$ from attacker $\mathcal{A}$ and simulator $\mathcal{S}$.

3 The Construction

Now we proceed to give the construction. Given subverted implementations of the hash functions $\{\tilde{h}_i\}_{i=0,\ldots,\ell}$, (the original version of each is $h_i(\cdot)$ could be considered as $h(i,\cdot)$), the corrected function is defined as:

$$\tilde{h}_R(x) = \tilde{h}_0(\tilde{g}_R(x)) = \tilde{h}_0\left(\bigoplus_{i=1}^{\ell} \tilde{h}_i(x \oplus r_i)\right).$$

where $R = (r_1, \ldots, r_\ell)$ are sampled uniformly after $\{\tilde{h}_i(\cdot)\}$ is provided, and then revealed to the public, and the internal function $\tilde{g}_R(\cdot)$ is defined below:

$$\tilde{g}_R(x) = \bigoplus_{i=1}^{\ell} \tilde{h}_i(x \oplus r_i).$$

We wish to show that such a construction will be indifferentiable to an actual random oracle (with the proper input/output length). This implies that the distribution of values taken by $\tilde{h}_R(\cdot)$ at inputs that have not been queried have negligible distance from the uniform distribution.

Theorem 3. *Suppose* $h : \{0,\ldots,\ell\} \times \{0,1\}^n \to \{0,1\}^n$ *defines a family of random oracles* $h_i : \{0,1\}^n \to \{0,1\}^n$ *as* $h(i,\cdot)$*, for* $i = 0,\ldots,\ell$*, and* $\ell \geq 3n+1$*. Consider a (subversion) algorithm* $\tilde{H}$ *and* $\tilde{H}^h(x)$ *defines a subverted random oracle* $\tilde{h}$*. Assume that for every* h *(and every* i*),*

$$\Pr_{x\in\{0,1\}^n}[\tilde{h}(i,x) \neq h(i,x)] = \mathsf{negl}(n)\,. \tag{1}$$

The construction $\tilde{h}_R(\cdot)$ *is* $(t_{\widehat{\mathcal{D}}}, t_S, q, \epsilon)$*-indifferentiable from a random oracle* $F : \{0,1\}^n \to \{0,1\}^n$*, for any* $t_{\widehat{\mathcal{D}}}$*, with* $t_S = \mathrm{poly}(q)$*,* $\epsilon = \mathsf{negl}(n)$ *and* q *is the number of queries made by the distinguisher* $\widehat{\mathcal{D}}$ *as in Definition 3.*

Roadmap for the proof. We first describe the simulator algorithm. The main challenge for the simulator is to ensure the consistency of two ways of generating the output values of $\tilde{h}_R(\cdot)$, (it could also be reconstructed by querying the original random oracle directly together with the subverted implementation $\tilde{h}$ to replace the potentially corrupted terms). The idea for simulation is fairly simple: for an input x, $F(x)$ would be used to program h_0 on input $\tilde{g}_R(x)$.

There are two obstacles that hinder the simulation: (1) for some x, h_0 has been queried on $\tilde{g}_R(x)$ before the actual programing step, thus the simulator has to abort; (2) the distinguisher queries on some input x such that $\tilde{g}_R(x)$ falls into the incorrect portion of inputs to $\tilde{h}_0$.

To bound the probability of these two events, we first establish the property of the internal function $\tilde{g}_R(\cdot)$ that no adversary can find an input value that falls into a small domain (or for any input x, the output is unpredicatable to the adversary if he has not made any related queries.). See Theorem 4 below. Note that the bound is conditioned on adaptive queries of the distinguisher.

Theorem 4 (Informal). *Suppose the subverted implementation disagrees with the original oracle at only a negligible fraction of inputs, then with an overwhelming probability in* R*, conditioned on the* $h(q_1),\ldots,h(q_s)$ *(made by any* $\widehat{\mathcal{D}}$*), for all* x *outside the "queried" set* $\{t \mid h_i(t \oplus r_i)$ *was queried*$\}$*, and every event* $E \subset \{0,1\}^n$*,*

$$\Pr_h[\tilde{g}_R(x) \in E] \leq \mathrm{poly}(n)\sqrt{\Pr[E]} + \mathsf{negl}(n).$$

In particular, if $|E|$ *is exponentially small in* $\{0,1\}^n$*, the probability* $\tilde{g}_R(x)$ *falls into* E *would be negligible for any* x*.*

Next, our major analysis will focus on proving this theorem for $\tilde{g}_R(\cdot)$.

We first set down and prove a "rejection resampling" lemma. This is instrumental in our approach to Theorem 5 (the formal version of Theorem 4), showing that this produces unpredictable values, even to an adaptive adversary with access to the (public) randomness R;

Surveying the proof in more detail, recall that a value $\tilde{g}_R(x)$ is determined as the XOR of a "constellation" of values $\bigoplus \tilde{h}_i(x \oplus r_i)$; intuitively, if we could be sure that **(a)** at least one of these terms, say $x \oplus r_i$, was not queried by $\tilde{H}^h(\cdot)$ when evaluated on the other terms and, **(b)** this isolated term $x \oplus r_i$ was answered "honestly" (that is, $\tilde{h}_i(x \oplus r_i) = h_i(x \oplus r_i)$), then it seems reasonable to conclude that the resulting value, the XOR of the results, is close to uniform.

However, applying this intuition to rigorously prove Theorem 5 faces a few challenges. Perhaps the principal difficulty is that it is not obvious how to "partially expose" the random oracle h to take advantage of this intuition: specifically, a traditional approach to proving such strong results is to expose the values taken by the $h(x)$ "as needed," maintaining the invariant that the unexposed values are uniform and conditionally independent on the exposed values.

In our setting, we would ideally like to expose all but one of the values of a particular constellation $\{h_i(x \oplus r_i)\}$ so as to guarantee that the last (unexposed) value has the properties (a) and (b) above. While randomly guessing an ordering could guarantee this with fairly high probability $\approx 1 - 1/\ell$ *we must have such a favorable event take place for all* x, and so must somehow find a way to guarantee exponentially small failure probabilities. The **rejection resampling lemma**, discussed above, permits us to examine all the points in a particular constellation, identify a good point (satisfying (a) and (b)) and then "pretend" that we never actually evaluated the point in question. In this sense, the resampling lemma quantifies the penalty necessary for "unexposing" a point of interest.

A less challenging difficulty is that, even conditioned on $\tilde{h}_i(x \oplus r_i) = h_i(x \oplus r_i)$, this value may not be uniform, as the adversary may choose to be "honest" based on some criteria depending on x or, even, other adaptively-queried points. Finally, of course, the subversion algorithm $\tilde{H}^h(\cdot)$ is fully-adaptive, and only needs to disrupt $\tilde{g}_R(x)$ at a single value of x.

4 Security Proof

We begin with an abstract formulation of the properties of our construction and the analysis, and then transition to the detailed description of the simulator algorithm and its effectiveness.

4.1 The Simulator Algorithm

The main task of the simulator is to ensure the answers to $\{h_i\}$-queries to be consistent with the value of $F(\cdot)$, since for each input x, $\tilde{h}_R(x)$ is determined by a sequence of related queries to $\{h_i\}$ and $\tilde{H}$, (or simply the backdoor z) and the value of R. The basic idea is to program the external layer h_0 using values of $F(x)$, such that the value $F(x)$ is set for $h_0(\tilde{g}_R(x))$. The value $\tilde{g}_R(x)$ is obtained by $\mathcal{S}$ executing the subverted implementations $\{\tilde{h}_i\}$.

Let us define the simulator $\mathcal{S}$ (answering queries in two stages) as below:

In the first stage, $\mathcal{A}$ makes random oracle queries when manufacturing the subverted implementations $\{\tilde{h}_i\}_{i=0,\ldots,\ell}$.

On input queries $x_1, \ldots, x_{q_1}$ (at $\mathcal{A}$'s choice on which random oracle to query) that $\mathcal{A}$ makes before outputting the implementations (and the backdoor), $\mathcal{S}$ answers all those using uniform strings respectively. $\mathcal{S}$ maintains a table. See Table 1. (w.l.o.g, we simply assume the adversary asks all the hash queries for each value x_i, if not, the simulator asks himself to prepare the table.) $\mathcal{S}$ and $\mathcal{D}$ also both receive a random value for R.

Table 1. RO queries in phase-I

RO query x_i	$h_0(x_i)$	$h_1(x_i)$	$\ldots$	$h_\ell(x_i)$
x_1	$v_{1,0}$	$v_{1,1}$	$\ldots$	$v_{1,\ell}$
x_2	$v_{2,0}$	$v_{2,1}$	$\ldots$	$v_{2,\ell}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ldots$
x_{q_1}	$v_{q_1,0}$	$v_{q_1,1}$	$\ldots$	$v_{q_1,\ell}$

In the second stage, the distinguisher $\mathcal{D}$ now having input R, will ask both queries to the construction and the random oracles. The simulator $\mathcal{S}$ now also has these extra information of R and oracle access to the implementation $\tilde{h}$ and will answer the random oracle queries to ensure consistency. In particular:

On input query m_j to the k_j-th random oracle h_{k_j}, $\mathcal{S}$ defines the adjusted query $m'_j := m_j \oplus r_{k_j}$, and prepares answers for all related queries, i.e., for each i, the input $m'_j \oplus r_i$ to h_i; and the input $\tilde{g}_R(m'_j)$ to h_0.

- If $k_j > 0$:
  $\mathcal{S}$ runs the implementation $\tilde{h}_i$ on $m'_j \oplus r_i = m_j \oplus r_{k_j} \oplus r_i$, for all $i \in \{1, \ldots, \ell\}$, to derive the value $\tilde{g}_R(m'_j) = \bigoplus_{i=1}^{\ell} \tilde{h}_i(m'_j \oplus r_i)$. During the execution of $\tilde{h}_i$ on those inputs, $\mathcal{S}$ also answers the random oracle queries (or read from Table 1) on those values if the implementation makes any. In more detail,
  1. $\mathcal{S}$ first checks in both tables whether $m'_j \oplus r_i$ has been queried for h_i (Table 2 first), if queried in either of them, $\mathcal{S}$ returns the corresponding answer; if not queried, $\mathcal{S}$ simply returns a random value $u_{j,i}$ as answer and records it in Table 2;
  2. $\mathcal{S}$ checks whether $\tilde{g}_R(m'_j)$ has been queried for h_0. If not, $\mathcal{S}$ queries F on m'_j and gets a response $F(m'_j)$. $\mathcal{S}$ then checks whether m'_j has been queried in stage-I, (i.e., check Table 1). If yes and the corresponding value $v_{j,0}$ does not equal to $F(m'_j)$, $\mathcal{S}$ aborts; otherwise, $\mathcal{S}$ sets $F(m'_j) = u_{j,0}$ as the answer for $h_0(\tilde{g}_R(m'_j))$.
- If $k_j = 0$:
  $\mathcal{S}$ checks whether m_j has been queried for h_0 in stage-II, i.e., there exists an m'_t in Table 2 such that $\tilde{g}_R(m'_t) = m_j$. If yes, $\mathcal{S}$ simply uses the corresponding value $u_{t,0}$ as answer; If not, $\mathcal{S}$ checks whether it has been queried in stage-I, $\mathcal{S}$ returns the value of $v_{i,0}$ if m_j has been queried. Otherwise, $\mathcal{S}$ chooses a random value $v_{j,0}$ as the response and records it in Table 2.

Table 2. Phase-II queries: The headers are adjusted random oracle queries $m'_i = m_i \oplus r_{k_i}$ if m_i is queried for h_{k_i}, and $u_{i,0} = F(m_i)$.

Adjusted query m'_i	$h_1(m'_i \oplus r_1)$	$\ldots$	$h_\ell(m'_i \oplus r_\ell)$	$\tilde{g}_R(m'_i)$	$h_0\left(\tilde{g}_R(m'_i)\right)$
$m'_1 = m_1 \oplus r_{k_1}$	$u_{1,1}$	$\ldots$	$u_{1,\ell}$	$\tilde{g}_R(m'_1)$	$u_{1,0}$
$m'_2 = m_2 \oplus r_{k_2}$	$u_{2,1}$	$\ldots$	$u_{2,\ell}$	$\tilde{g}_R(m'_2)$	$u_{2,0}$
$\vdots$	$\vdots$	$\ldots$	$\vdots$	$\vdots$	$\vdots$
$m'_q = m_q \oplus r_{k_q}$	$u_{q,1}$	$\ldots$	$u_{q,\ell}$	$\tilde{g}_R(m'_q)$	$u_{q,0}$

Probability analysis. Let us define the event that $\mathcal{S}$ aborts as Abort. According to the description, $\mathcal{S}$ aborts only when the distinguisher $\mathcal{D}$ finds an input m such that the $\tilde{g}_R(m) = x$ has either been queried for h_0 in stage-I, or queried in stage-II before any of $\{m \oplus r_i\}_{i=1,\ldots,\ell}$ has been queried for h_i. We can define $T_0 = \{x | x \text{ is queried for } h_0\}$, and following Theorem 5 (to be proven below), $\Pr[\mathsf{Abort}] \leq \frac{|T_0|}{2^n} \leq \frac{q+q_1}{2^n} \leq \mathsf{negl}(n)$ for any polynomially large q, q_1.

We also define the event Bad as that the distinguisher finds an input m, such that $\tilde{h}_0(\tilde{g}_R(m)) \neq h_0(\tilde{g}_R(m))$, also we define $T_1 = \{m | \tilde{h}_0(\tilde{g}_R(m)) \neq h_0(\tilde{g}_R(m))\}$. Following Theorem 5, $\Pr[\mathsf{Bad}] \leq \frac{|T_1|}{2^n} + \mathsf{negl}(n) \leq \mathsf{negl}(n)$. The latter inequality comes from the condition $\tilde{h}$ disagrees with h only at negligible fraction of inputs.

Furthermore, it is easy to see, conditioned on $\mathcal{S}$ does not abort, and Bad does not happen, the simulation is perfect.

In the rest of the paper, we will focus on proving our main theorem about the property of the internal function $\tilde{g}_R(\cdot)$.

4.2 A Rejection Resampling Lemma

We first prove a general rejection re-sampling lemma, and use it as a machinery to prove our main theorem for $\tilde{g}_R(\cdot)$. Let $\Omega_1, \ldots, \Omega_k$ be a family of sets and let $\Omega = \Omega_1 \times \cdots \times \Omega_k$. We treat Ω as a probability space under the uniform probability law: for an event $E \subset \Omega$, we let $\mu(E) = |E|/|\Omega|$ denote the probability of E. For an element $x = (x_1, \ldots, x_k) \in \Omega$ and an index i, we define the random variable $R_i x = (x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_k)$ where y is drawn uniformly at random from Ω_i. We say that such a random variable arises by "resampling" x at the index i.

We consider the effect that arbitrary "adversarial" resampling can have on the uniform distribution. Specifically, for a function $A : \Omega \to \{1, \ldots, k\}$, we consider the random variable $R_{A(X)}X$, where X is a uniformly distributed random variable and the index chosen for resampling is determined by A (as a function of X). By this device, the function A implicitly defines a probability law μ_A on Ω, where the probability of an event E is given by

$$\mu_A(E) = \Pr[R_{A(X)}X \in E] .$$

Lemma 1 (Rejection re-sampling). *Let X be a random variable uniform on $\Omega = \Omega_1 \times \cdots \times \Omega_k$. Let $A : \Omega \to \{1, \ldots, , k\}$ and define $Z = R_{A(X)}X$ and μ_A as above. Then, for any event E,*

$$\frac{\mu(E)^2}{k} \leq \mu_A(E) \leq k \cdot \mu(E)\,.$$

Remark 4. Jumping ahead, such a resampling lemma will be used to define a good event E such that one term of $\tilde{h}_i(x \oplus r_i)$ will be uniformly chosen (not correlated with any other term), thus yields a uniform distribution for the summation. The actual adversarial distribution $\mu_A(E)$ is thus bounded not too far from $\mu(E)$. Let us first prove this useful lemma.

Proof. Consider an event $E \subset X$. To simplify our discussion of the adversarial resampling process discussed above, we remark that the random variables R_iX and $R_{A(X)}X$ can be directly defined over the probability space $\Omega \times \Omega$: Consider two independent random variables, X and Y, each drawn uniformly on Ω; then, for any i the random variable R_iX can be described $(X_1, \ldots, X_{i-1}, Y_i, X_{i+1}, \ldots, X_k)$ and $R_{A(X)}X = (Z_1, \ldots, Z_k)$ where

$$Z_i = \begin{cases} Y_i & \text{if } i = A(X), \\ X_i & \text{otherwise.} \end{cases}$$

Note that for any fixed i, the probability law of R_iX is the uniform law on Ω.

Upper bound. It follows that for an event E

$$\mu_A(E) = \Pr_{X,Y}[R_{A(X)}X \in E] \leq \Pr_{X,Y}[\exists i, R_iX \in E] \leq k \cdot \mu(E)\,,$$

which establishes the claimed upper bound on $\mu_A(E)$.

Lower bound. As for the lower bound, define

$$B_i = \{x \in \Omega \mid A(x) = i\} \qquad \text{and} \qquad E_i = E \cap B_i\,.$$

As the B_i, E_i partition Ω, E respectively, and $\sum_i \mu_A(E_i) = \mu_A(E)$. Observe that

$$\begin{aligned} \Pr_{X,Y}[R_{A(X)}X \in E] = \sum_i \Pr_{X,Y}[R_{A(x)}X \in E_i] &\geq \sum_i \Pr_{X,Y}[X \in B_i \text{ and } R_iX \in E_i] \\ &\geq \sum_i \Pr_{X,Y}[X \in E_i \text{ and } R_iX \in E_i]\,. \end{aligned} \tag{2}$$

To complete the proof, we will prove that for any i and for any event F

$$\Pr_{X,Y}[X \in F \,\text{and}\, R_iX \in F] \geq \Pr[F]^2\,. \tag{3}$$

Putting aside the proof of (3) for a moment, observe that applying (3) to the events E_i in the expansion (2) above yields the following by Cauchy-Schwarz.

$$\Pr_{X,Y}[R_{A(X)}X \in E] \geq \sum_i \Pr[E_i]^2 \geq \frac{\Pr[E]^2}{k}$$

Finally, we return to establish (3). Observe that for an event F,

$$\Pr_{X,Y}[X \in F \text{ and } R_i X \in F]$$

$$= \frac{1}{|\Omega_i|} \sum_{(x_1,\ldots,x_k)\in\Omega} \Pr[X \in F \text{ and } R_i X \in F \mid \forall j \neq i, X_j = x_j] \Pr[\forall j \neq i, X_j = x_j] .$$

(The leading $1/|\Omega_i|$ term cancels the sum over x_i, which not referenced in the argument of the sum.) Under such strong conditioning, however, the two events $X \in F$ and $R_i X \in F$ are independent and, moreover, have the same probability. (Conditioned on the other coordinates, the event depends only on coordinate i of the result that is uniform and independent for the two random variables.) As

$$\Pr[\forall j \neq i, X_j = x_j] = \frac{1}{\prod_{i\neq j} |\Omega_j|}$$

we may rewrite the sum above as

$$\begin{aligned}
&\Pr_{X,Y}[X \in F \text{ and } R_i X \in F] \\
&= \frac{1}{|\Omega_i|} \sum_{(x_1,\ldots,x_k)\in\Omega} \Pr[X \in F \text{ and } R_i X \in F \mid \forall j \neq i, X_j = x_j] \cdot \frac{1}{\prod_{i\neq j} |\Omega_j|} \\
&= \frac{1}{|\Omega|} \sum_{(x_1,\ldots,x_k)\in\Omega} \Pr[X \in F \text{ and } R_i X \in F \mid \forall j \neq i, X_j = x_j] \\
&= \frac{1}{|\Omega|} \sum_{(x_1,\ldots,x_k)\in\Omega} \Pr[X \in F \mid \forall j \neq i, X_j = x_j]^2 \\
&\geq \frac{1}{|\Omega|^2} \left(\sum_{(x_1,\ldots,x_k)\in\Omega} \Pr[X \in F \mid \forall j \neq i, X_j = x_j] \right)^2 = \Pr[X \in F]^2 ,
\end{aligned}$$

where the inequality is Cauchy-Schwarz. □

We remark that these bounds are fairly tight. For the lower bound—the case of interest in our applications—let $E_i \subset \Omega_i$ be a family of events with small probability ϵ and $E = \{(\omega_1, \ldots, \omega_k, \omega_{k+1}) \mid \exists \text{ unique } i \leq k, \omega_i \in E_i\} \subset \prod_i^{k+1} \Omega_i$. When $\epsilon \ll 1/k$, $\Pr[E] \approx k\epsilon$ while $\Pr[R_{A(X)} X \in E] \approx k\epsilon^2 = (k\epsilon)^2/k$ for the strategy which, in case the event occurred, redraws the offending index and, in case the event did not occur, redraws the $k+1$st "dummy" index. For the upper bound, consider an event E consisting of a single point x in the hypercube $\{0,1\}^k$; then $\Pr[E] = 2^{-k}$ and $\Pr[R_{A(X)} X \in E] \geq 2^{-k}(k+1)/2$ for the strategy which re-randomizes any coordinate on which the sample and x disagree (the strategy can be defined arbitrarily on the point x itself).

4.3 Establishing Pointwise Unpredictability

In this section, we focus our attention on the "internal" function (for $\ell > 3n$)

$$\tilde{g}_R(x) = \bigoplus_{i=1}^{\ell} \tilde{h}_i(x \oplus r_i)\,.$$

In particular, we will prove that for each x, the probability that the adversary can force the output of $\tilde{g}_R(x)$ to fall into some range E is polynomial in the density of the range (that is, the probability that a uniform element lies in E). Thus, the output will be unpredictable to the adversary if she has not queried the corresponding random oracles.

Intuition about the analysis. As discussed above, we want to show that for every x, there exist at least one term $h_i(x \oplus r_i)$ satisfying: (i) $h_i(x \oplus r_i)$ is answered honestly (that is, $h_i(x \oplus r_i) = \tilde{h}_i(x \oplus r_i)$; (ii) $h_i(x \oplus r_i)$ is not correlated with other terms. In order to ensure condition (ii), we proceed in two steps. We first turn to analyze the probability that $h_i(x \oplus r_i)$ has not been queried by $H^{h_*}(x \oplus r_j)$ for all other index j. This is still not enough to demonstrate perfect independence, as the good term is subject to the condition of (i), but it suffices for our purposes. As discussed above, for analytical purposes we consider an exotic distribution that calls for this "good" term to be independently re-sampled and apply rejection re-sampling lemma to ensure the original (adversarial) distribution is not too far from the exotic one. We first recall the theorem for the internal function:

Theorem 5. *Suppose $h : \{0,\ldots,\ell\} \times \{0,1\}^n \to \{0,1\}^n$ defines a family of random oracles $h_i : \{0,1\}^n \to \{0,1\}^n$ as $h(i,\cdot)$, for $i = 0,\ldots,\ell$, and $\ell \geq 3n+1$. Consider a (subversion) algorithm H and $H^h(x)$ defines a subverted random oracle $\tilde{h}$. Assume that for every h (and every i),*

$$\Pr_{x \in \{0,1\}^n}[\tilde{h}(i,x) \neq h(i,x)] = \mathsf{negl}(n)\,.$$

Then, with overwhelming probability in R, h, and conditioned on the $h(q_1),\ldots,h(q_s)$ (made by any $\widehat{\mathcal{D}}$), for all x outside the "queried" set $\{t \mid h_i(t \oplus r_i)$ was queried$\}$ and every event $E \subset \{0,1\}^n$,

$$\Pr_h[\tilde{g}_R(x) \in E] \leq \mathrm{poly}(n)\sqrt{\Pr[E]} + \mathsf{negl}(n).$$

Proof. Throughout the estimates, we will assume that $\ell > 3n$. Here, we overload the notation h_* to denote the collection of functions $h_1,\ldots,h_\ell$.

We begin by considering the simpler case where no queries are made, and just focus on controlling the resulting values vis-a-vis a particular event E. At the end of the proof, we explain how to handle an adaptive family of queries.

Guaranteeing honest answers. First, we ensure that with high probability in R and h_*, for every x, there is a contributing term $\tilde{h}_i(x \oplus r_i)$ that is likely (if the random variable $h_i(x \oplus r_i)$ is *redrawn* according to the uniform distribution)

to be "honest" in the sense that $\tilde{h}_i(x \oplus r_i) = h_i(x \oplus r_i)$. The reason that this simple property does not follow straightforwardly is due to the fact that $\tilde{h}_i$ may adaptively define the "dishonest" points which are not fixed during the manufacturing of $\tilde{h}_i$.

To begin, let us consider the following random variables defined by random selection of h_* (denoting the $\{h_i\}$) and R, (later used to bound the number of dishonest terms):

$$d_i(\alpha) = \begin{cases} 1 & \text{if } \tilde{h}_i(\alpha) \neq h_i(\alpha), \\ 0 & \text{otherwise;} \end{cases} \qquad \text{and} \qquad D_i(\alpha) = \mathbb{E}_{h_i(\alpha)}[d_i(\alpha)]\,.$$

(Throughout $\mathbb{E}[\cdot]$ denotes expectation. For a given h_* and an element α, the value $D_i(\alpha)$ is defined by redrawing the value of $h_i(\alpha)$ uniformly at random; equivalently, $D_i(\alpha)$ is the conditional expectation of $d_i(\alpha)$ obtained by setting all other values of $h_*()$ except $h_i(\alpha)$.) Note that by assumption, for each i,

$$\mathbb{E}_{h_*}\,\mathbb{E}_\alpha[D_i(\alpha)] = \mathbb{E}_{h_*}\,\mathbb{E}_\alpha\,\mathbb{E}_{h_i(\alpha)}[d_i(\alpha)] = \mathbb{E}_{h_*}\,\mathbb{E}_\alpha[d_i(\alpha)] \leq \epsilon\,,$$

where α is chosen uniformly and ϵ is the (negligible) disagreement probability of (1) above.

We introduce several events that play a basic role in the proof.

- Flat functions. We say that h_* is *flat* if, for each i, $\mathbb{E}_\alpha[D_i(\alpha)] \leq \epsilon^{1/3}$, where α is drawn uniformly.

Note that $\Pr[h_* \text{ not flat}] = \Pr[\exists i \in [\ell], \mathbb{E}_\alpha[D_i(\alpha)] > \epsilon^{1/3}]$, thus

$$\Pr[h_* \text{ not flat}] \leq \ell \cdot \mathbb{E}_{h_*}\,\mathbb{E}_\alpha[D_i(\alpha)]/\epsilon^{1/3} = \ell\epsilon^{2/3}$$

by Markov's inequality and the union bound. Further, observe that if h_* is flat, then for any $x \in \{0,1\}^n$, any $0 < k \leq \ell$, and random choices of $R = \{r_1, \ldots, r_\ell\}$,

$$\mathbb{E}_R \sum_{\substack{I \subset [\ell], \\ |I|=k}} \prod_i D_i(x \oplus r_i) = \sum_{\substack{I \subset [\ell], \\ |I|=k}} \prod_i \mathbb{E}_{r_i}\, D_i(x \oplus r_i) \leq \binom{\ell}{k} \epsilon^{k/3} \leq (\ell^3\epsilon)^{k/3}\,.$$

Next, we will use this property to show that with a sufficiently large ℓ, e.g., $\ell = 3n$, then for each x, we can find an index i such that $D_i(x \oplus r_i)$ is small.

- Honesty under resampling. For a tuple $R = (r_1, \ldots, r_\ell)$, functions h_*, and an element $x \in \{0,1\}^n$, we say that the triple (R, h_*, x) is *honest* if

$$\sum_{\substack{I \subset [\ell], \\ |I|=3n}} \prod_i D_i(x \oplus r_i) \leq 2^{3n}(\ell^3\epsilon)^n\,.$$

If R and h_* are "universally" honest, which is to say that (R, h_*, x) is honest for all $x \in \{0,1\}^n$, we simply say (R, h_*) is honest. Then $\Pr_R[(R, h_*) \text{ is not honest}] = \Pr_R[\exists x, (R, h_*, x) \text{ is not honest}]$. When h_* is flat, we have the following:

$$\Pr_R[(R, h_*) \text{ is not honest}] \leq 2^n \cdot \mathbb{E}_R \left(\sum_{\substack{I \subset [\ell], \\ |I|=3n}} \prod_i D_i(x \oplus r_i) \right) / 2^{3n}(\ell^3\epsilon)^n \leq 2^{-2n}$$

by Markov's inequality (on the random variable $\sum_{\substack{I\subset[\ell],\\ |I|=3n}} \prod_i D_i(x\oplus r_i)$) and the union bound. Observe that if (R, h_*) is honest, then for every x,

$$\max_{\substack{I\subset[\ell]\\ |I|=3n}} \prod_i D_i(x\oplus r_i) \le 2^{3n}(\ell^3\epsilon)^n\,.$$

It follows that, for every set I of size $3n$, there exists an element $i \in I$ so that

$$D_i(x\oplus r_i) \le \sqrt[3n]{2^{3n}(\ell^3\epsilon)^n} = 2\ell\sqrt[3]{\epsilon}.$$

That said, *conditioned on h_* being flat*, with an overwhelming probability (that is, $1-\mathsf{negl}(n)$) in R, the pair (R, h_*) is honest and so gives rise to at least one small $D_i(x\oplus r_i)$ for each x (recall that the smaller $D_i(\alpha)$ is, the fewer points that h_i disagrees $\tilde{h}_i$).

Unfortunately, merely ensuring that some term of each "constellation" $\{\tilde{h}_i(x\oplus r_i)\}$ is honest with high probability is not enough—it is possible that a clever adversary can adapt other terms to an honest term to interfere with the final value of $\tilde{g}_R(x)$. The next part focuses on controlling these dependencies.

Controlling dependence among the terms. We now transition to controlling dependence between various values of $\tilde{h}_i(x)$. In particular, for every x we want to ensure that there exists some i so that $h_i(x\oplus r_i)$ was never queried by $\tilde{H}^{h_*}(\cdot)$ when evaluated on all other $x\oplus r_j$, i.e., for all $j\in[\ell]$ and $j\neq i$.

Note that the set of queries made by $\tilde{H}^{h_*}(u)$ is determined entirely by h_* and u: thus, conditioned on a particular h_*, the event (over R) that $\tilde{H}^{h_*}(u)$ queries $h_s(x\oplus r_s)$ and the event that $\tilde{H}^{h_*}(u')$ queries $h_t(x\oplus r_t)$ are independent (for any u, u' and $s\neq t$). We introduce the following notation for these events: for a pair of indices i, j $(i\neq j)$, we define

$$Q_{i\to j}(x) = \begin{cases} 1 & \text{if } \tilde{H}^{h_*}(x\oplus r_i) \text{ queries } h_j(x\oplus r_j),\\ 0 & \text{otherwise.}\end{cases}$$

In light of the discussion above, for an element x and a fixed value of h_*, consider a subset $T\subset[\ell]$ and a function $s: T\to[\ell]$; we treat such a function as a representative for the event that each "target" $t\in T$ was queried by a "source" $s(t)$. (Note that there could be multiple such functions $s(\cdot)$ for T). We define

$$Q_s(x) = \prod_{t\in T} Q_{s(t)\to t}(x)\,.$$

We also introduce a couple of new notions for the ease of presentation:

- Independent fingerprints. We say that a representative $s: T\to[\ell]$ is *independent* if $s(T)\cap T = \emptyset$, which is to say that the range of the function lies in $[\ell]\setminus T$. For any such *independent* fingerprint $s(\cdot)$, note that (for any x, h_*):

$$\mathbb{E}_R[Q_s(x)] = \mathbb{E}_R\left[\prod_{t\in T} Q_{s(t)\to t}(x)\right] = \prod_{t\in T}\mathbb{E}_R\left[Q_{s(t)\to t}(x)\right] \le \left(\frac{\tau(n)}{2^n}\right)^{|T|}, \quad (4)$$

where $\tau(n)$ denotes the running time (and, hence, an upper bound on the number of queries) of $\tilde{H}^h(x)$ on inputs of length n.

We will next use such notion to bound the number of bad terms that were queried by some other term.

- Dangerous set. For a fixed x, h_*, and R, we say that a set $T \subset [\ell]$ is *dangerous* if every element t in T is queried by some $\tilde{H}^{h_*}(x \oplus r_i)$ for $i \neq t$.

We claim that if T is a dangerous set then we can always identify an *independent* fingerprint $s : T' \to [\ell]$ for a subset $T' \subset T$ with $|T'| \geq |T|/2$.

To see this, we build T' as follows: write $T = \{t_1, \ldots, t_m\}$ and consider the elements in order $t_1, \ldots, t_m$; for each element t_i, we add it to T', if t_i is queried by some elements in T[4], pick one of them, denoted as t_j (for $j > i$), define $s(t_i) = t_j$, and remove t_j from T. Observe now that (i) each element in T' maps to a value (or was queried by a term) outside of T'; (ii) each element t_i added to T' removes at most two elements of T (t_i and $s(t_i)$), and hence $|T'| \geq |T|/2$.

If follows that for a set T the number of such possible independent fingerprints (whose image is at least half the size of T) is bounded by:

$$\sum_{m \geq |T|/2} \binom{|T|}{m} (\ell - 1) \cdots (\ell - m) \leq 2^{|T|} \ell^{|T|}$$

We conclude from (4) that for any fixed set T (and any fixed x and h_*)

$$\begin{aligned} \Pr_R[T \text{ is dangerous}] \leq \quad & \Pr[Q_s(x) \text{ occurs for some independent fingerprint}] \\ \leq \quad & 2^{|T|} \ell^{|T|} \left(\frac{\tau(n)}{2^n} \right)^{|T|/2} = \left(\frac{4\ell^2 \tau(n)}{2^n} \right)^{|T|/2}. \end{aligned}$$

By taking the union bound over all sets T of size k, it follows immediately that

$$\begin{aligned} &\Pr_R\left[k \text{ of the } h_i(x \oplus r_i) \text{ are queried by some other } \{\tilde{h}_j(x \oplus r_j)\}_{j \neq i, j \in [\ell]}\right] \\ &\leq \binom{\ell}{k} \left(\frac{4\ell^2 \tau(n)}{2^n} \right)^{k/2} \leq \ell^k \left(\frac{4\ell^2 \tau(n)}{2^n} \right)^{k/2} \leq \left(\frac{4\ell^4 \tau(n)}{2^n} \right)^{k/2}. \end{aligned} \tag{5}$$

The above bound guarantees that, for any fixed x, with overwhelming probability, there are $\ell - k$ terms that were never queried by *any* other terms.

- k-sparsity: Finally, we say that the pair (R, h_*) is *k-sparse* if for all x, the set of queries made by $H^{h_*}(x \oplus r_i)$ includes no more than k of the $h_i(x \oplus r_i)$.

Applying the union bound over all 2^n strings x to (5), we conclude that, for even constant k (say $k = 5$), we have

$$\Pr_{h_*, R}[(R, h_*) \text{ is not } k\text{-sparse}] \leq 2^n \left(\frac{4\ell^4 \tau(n)}{2^n} \right)^{k/2} \leq 2^{-n^{\Theta(k)}}.$$

[4] We overload the notation a bit, here the elements in T simply denote the indices of the terms.

With the preparatory work behind us, we turn to guaranteeing that each x possesses a good term (one that is both well behaved under resampling and not queried by other terms).

Establishing existence of a good term. Next, we wish to show that for any event E with $\Pr[E] = \mu(E)$, and for any x,

$$\Pr_{h_*,R}[\tilde{g}_R(x) \in E] \leq \mathrm{poly}(n)\sqrt{\mu(E)} + \mathsf{negl}(n)\,.$$

In particular, if E has negligible density, the probability that $\tilde{g}_R(x) \in E$ is likewise negligible.

We say that R is *flat-honest* if $\Pr_{h_*}[(R, h_*) \text{ not honest} \mid h_* \text{ is flat}] \leq 2^{-n}$. Observe that by Markov's inequality a uniformly selected R is flat-honest with probability $1 - 2^{-n}$.

We also say R is *uniformly-k-sparse* if $\Pr_{h_*}[(R, h_*) \text{ is not} k - \text{sparse}] \leq 2^{-n}$. Assuming k is a sufficiently large constant (e.g., 5), note that by Markov's inequality a random R is uniformly-k-sparse with probability $1 - 2^{-n}$.

Now we know that selection of a uniformly random $R = (r_1, \ldots, r_\ell)$, with probability $1 - 2^{n-1}$, is *both* uniformly-k-sparse (for the constant k discussed above) and flat-honest. We condition, for the moment, on such a choice of R. In this case, a random function h_* is likely to be *both k-sparse and honest*: it follows that for every x there is some term $h_i(x \oplus r_i)$ that is not queried by H to determine the value of the other terms and, moreover, it is equal to $\tilde{h}_i(x \oplus r_i)$ with an overwhelming probability. We say that such a pair (R, h_*) is *unpredictable*; otherwise, we say that (R, h_*) is predictable.

$$\begin{aligned}
\Pr_{h_*}[h_* \text{ predictable (for } R)] &= \Pr_{h_*}[(R, h_*) \text{ not } k\text{-sparse or not honest}] \\
&\leq \Pr_{h_*}[(R, h_*) \text{ not } k\text{-sparse}] + \\
&\quad \Pr_{h_*}[(R, h_*) \text{ not honest and } h_* \text{ flat}] + \\
&\quad \Pr_{h_*}[(R, h_*) \text{ not honest and } h_* \text{ not flat}] \\
&\leq \Pr_{h_*}[(R, h_*) \text{ not } k\text{-sparse}] + \\
&\quad \Pr_{h_*}[(R, h_*) \text{ not honest} \mid h_* \text{ flat}] + \\
&\quad \Pr_{h_*}[h_* \text{ not flat}] \\
&\leq 2^{-n} + 2^{-n} + \ell\epsilon^{2/3}\,.
\end{aligned} \tag{6}$$

For each x, we can be sure there is at least one term $\tilde{h}_i(x \oplus r_i)$ which is typically answered according to $h_i(x \oplus r_i)$ (i.e., answered honestly) and never queried by the other terms. Unfortunately, to identify this term, we needed to evaluate $\tilde{h}$ on the whole constellation of points; the rejection resampling lemma lets us correct for this with a bounded penalty.

To complete the analysis, we consider the following experiment: conditioned on R, consider the probability that $\tilde{g}_R(x) \in E$ when h_* is drawn as follows:

– if (R, h_*) is unpredictable, there is a (lexicographically first) index i for which $h_i(x \oplus r_i)$ is queried by no other $\tilde{h}_j(x \oplus r_j)$ and is honest. Now, *redraw* the value of $h_i(x \oplus r_i)$ uniformly at random.

These rules define a distribution on h_* that is no longer uniform. Note, however, that redrawing $h_i(x \oplus r_i)$ does not affect the values of $\tilde{h}_i(x \oplus r_j)$ (for distinct j); as $\tilde{g}_R(x) = \bigoplus_i \tilde{h}_i(x \oplus r_i)$, under this exotic distribution (for any x),

$$\Pr_{\text{resampled } h_*}\left[\tilde{g}_R(x) \in E \,\middle|\, \begin{matrix} R \text{ unif. } k\text{-sparse} \\ \& \text{ flat honest} \end{matrix}\right] \leq \mu(E) + (2 \cdot 2^{-n} + \ell\epsilon^{2/3}) + 2\ell\epsilon^{1/3}\,,$$

where the $2\ell\epsilon^{1/3}$ term arises because we have only the guarantee that $D_i(x) \leq 2\ell\epsilon^{1/3}$ from the condition on honesty.

However, based on the **rejection resampling lemma** above, we conclude

$$\begin{aligned} \Pr_{h_*}\left[\tilde{g}_R(x) \in E \,\middle|\, \begin{matrix} R \text{ unif. } k\text{-sparse} \\ \& \text{ flat honest} \end{matrix}\right] &\leq \sqrt{\ell\left(\mu(E) + 2 \cdot 2^{-n/2} + 3\ell\epsilon^{1/3}\right)} \\ &\leq O(\sqrt{\ell\mu(E)} + \sqrt{\ell}2^{-n/4} + \ell\epsilon^{1/6})\,. \end{aligned} \tag{7}$$

and, hence, that

$$\begin{aligned} \Pr_{h_*,R}[\tilde{g}_R(x) \in E] &\leq 2^{-n} + O\left(\sqrt{\ell\mu(E)} + \sqrt{\ell}2^{-n/4} + \ell\epsilon^{1/6}\right) \\ &= O\left(\sqrt{\ell\mu(E)} + \sqrt{\ell}2^{-n/4} + \ell\epsilon^{1/6}\right)\,, \end{aligned}$$

where the 2^{-n} term comes from the cases that a randomly chosen R is not flat-honest or universal-k-sparse.

Conditioning on adaptive queries. Finally, we return to the problem of handling adaptive queries. With R and z fixed, the queries generated by $Q^{h_*}(R, z)$ depend only on h_* and we may ramify the probability space of h according to the queries and responses of Q; we say $\alpha = ((q_1, a_1), \ldots, (q_t, a_t))$ is a *transcript* for Q if Q queries h_* at $q_1, \ldots, q_t$ and receives the responses $a_1, \ldots, a_t$. We remark that if E is an event for which $\Pr_h[E \mid R, z] \leq \epsilon$, then by considering the natural martingale given by iterative exposure of values of h_* at the points queried by $Q^h(R, z)$, we have that $\epsilon \geq \Pr[E \mid R, z] = \sum_\alpha \Pr[E \mid \alpha, R, z] \cdot \Pr[\alpha \mid R, z]$. In particular, events with negligible probability likewise occur with negligible probability for all but a negligible fraction of transcripts α. Thus, the global properties of h discussed in the previous proof are retained even conditioned on a typical transcript α.

We require one amplification of the high-probability structural statements developed above. Note that, with overwhelming probability in R and h, every constellation $\{x \oplus r_i\}$ (an argument to h_i) contains only a constant number of points that are queried by more than a 2^{-n} fraction of other points in the domain

of h_*. (Indeed, the fraction of points in the domain of h_* that are queried by $H^h(z,x)$ for at least $w(n)$ values of x can be no more than $\text{poly}(n)/w(n)$, where the polynomial is determined by the running time of H.) We say that a pair R, z is *diffuse* if a randomly selected h has this property with probability $1 - 2^{-n/2}$; note that a random pair (R, z) is diffuse with probability $1 - 2^{-n/2}$.

Consider then conditioning on the event that (R, z) is flat-honest, uniformly-4-sparse, and diffuse; note that in this case, with high probability in h every x has an member of its constellation which is not queried by other members of the constellation, only queried by $H()$ at a vanishing fraction of other points in the domain, and has $D_i(x \oplus r_i) \leq 2\ell\sqrt[3]{\epsilon}$. We emphasize that these properties are global properties, holding for all x in the domain of h. In particular, we can apply the argument above to any x for which none of the q_i touch its constellation $\{x \oplus r_i\}$. This concludes the proof. □

5 Conclusions

We initiate the study of correcting subverted random oracles, where each subverted version disagrees with the original random oracle at a negligible fraction of inputs. We demonstrate that such an attack is devastating in several real-world scenarios. We give a simple construction that can be proven indifferentiable from a random oracle. Our analysis involves developing a new machinery of rejection resampling lemma which may be with independent interests. Our work provides a general tool to transform a buggy implementation of random oracle into a well-behaved one which can be directly applied to the kleptographic setting.

There are many interesting problems worth further exploring, such as better constructions, correcting other ideal objectives under subversion and more.

Acknowledgement. The authors thank Jonathan Katz for suggesting the indifferentiability framework as a modeling tool, and we thank anonymous reviewers for valuable comments.

References

1. Abelson, H., et al.: Keys under doormats. Commun. ACM **58**(10), 24–26 (2015)
2. Ateniese, G., Magri, B., Venturi, D.: Subversion-resilient signature schemes. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 15, pp. 364–375. ACM Press, October 2015
3. Bellare, M., Hoang, V.T.: Resisting randomness subversion: fast deterministic and hedged public-key encryption in the standard model. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 627–656. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_21
4. Bellare, M., Hoang, V.T., Keelveedhi, S.: Instantiating random oracles via UCEs. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 398–415. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_23
5. Bellare, M., Jaeger, J., Kane, D.: Mass-surveillance without the state: strongly undetectable algorithm-substitution attacks. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 15, pp. 1431–1440. ACM Press, October 2015

6. Bellare, M., Paterson, K.G., Rogaway, P.: Security of symmetric encryption against mass surveillance. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 1–19. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_1
7. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 93, pp. 62–73. ACM Press, Nov. (1993)
8. Bellovin, S.M., Blaze, M., Clark, S., Landau, S.: Going bright: wiretapping without weakening communications infrastructure. IEEE Secur. Priv. **11**(1), 62–72 (2013)
9. Blum, M.: Designing programs that check their work. Technical report TR-88-009, International Computer Science Institute, November 1988. http://www.icsi.berkeley.edu/pubs/techreports/tr-88-009.pdf
10. Blum, M., Kannan, S.: Designing programs that check their work. In: 21st ACM STOC, pp. 86–97. ACM Press, May 1989
11. Blum, M., Luby, M., Rubinfeld, R.: Self-testing/correcting with applications to numerical problems. In: 22nd ACM STOC, pp. 73–83. ACM Press, May 1990
12. Boldyreva, A., Cash, D., Fischlin, M., Warinschi, B.: Foundations of non-malleable hash and one-way functions. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 524–541. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_31
13. Boldyreva, A., Fischlin, M.: Analysis of random oracle instantiation scenarios for OAEP and other practical schemes. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 412–429. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_25
14. Boldyreva, A., Fischlin, M.: On the security of OAEP. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 210–225. Springer, Heidelberg (2006). https://doi.org/10.1007/11935230_14
15. Camenisch, J., Drijvers, M., Lehmann, A.: Anonymous attestation with subverted TPMs. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part III. LNCS, vol. 10403, pp. 427–461. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63697-9_15
16. Canetti, R.: Towards realizing random oracles: hash functions that hide all partial information. In: Kaliski, B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 455–469. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0052255
17. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: 42nd FOCS, pp. 136–145. IEEE Computer Society Press, October 2001
18. Canetti, R., Dakdouk, R.R.: Extractable perfectly one-way functions. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 449–460. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70583-3_37
19. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited (preliminary version). In: 30th ACM STOC, pp. 209–218. ACM Press, May 1998
20. Canetti, R., Micciancio, D., Reingold, O.: Perfectly one-way probabilistic hash functions (preliminary version). In: 30th ACM STOC, pp. 131–140. ACM Press, May 1998
21. Checkoway, S., et al.: A systematic analysis of the Juniper Dual EC incident. In: Proceedings of ACM CCS 2016 (2016). http://eprint.iacr.org/2016/376
22. Checkoway, S., et al.: On the practical exploitability of dual EC in TLS implementations. In: Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, 20–22 August 2014, pp. 319–335 (2014)

23. Coron, J.-S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård revisited: how to construct a hash function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_26
24. Coron, J.-S., Holenstein, T., Künzler, R., Patarin, J., Seurin, Y., Tessaro, S.: How to build an ideal cipher: the indifferentiability of the Feistel construction. J. Cryptol. **29**(1), 61–114 (2016)
25. Dachman-Soled, D., Katz, J., Thiruvengadam, A.: 10-round Feistel is indifferentiable from an ideal cipher. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 649–678. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_23
26. Dai, Y., Steinberger, J.: Indifferentiability of 8-round Feistel networks. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 95–120. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_4
27. Degabriele, J.P., Farshim, P., Poettering, B.: A more cautious approach to security against mass surveillance. In: Leander, G. (ed.) FSE 2015. LNCS, vol. 9054, pp. 579–598. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48116-5_28
28. Degabriele, J.P., Paterson, K.G., Schuldt, J.C.N., Woodage, J.: Backdoors in pseudorandom number generators: possibility and impossibility results. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 403–432. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_15
29. Demay, G., Gaži, P., Hirt, M., Maurer, U.: Resource-restricted indifferentiability. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 664–683. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_39
30. Dodis, Y., Ganesh, C., Golovnev, A., Juels, A., Ristenpart, T.: A formal treatment of backdoored pseudorandom generators. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 101–126. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_5
31. Dodis, Y., Guo, S., Katz, J.: Fixing cracks in the concrete: random oracles with auxiliary input, revisited. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part II. LNCS, vol. 10211, pp. 473–495. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56614-6_16
32. Dodis, Y., Mironov, I., Stephens-Davidowitz, N.: Message transmission with reverse firewalls–secure communication on corrupted machines. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. Part I, volume 9814 of LNCS, pp. 341–372. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_13
33. Dodis, Y., Puniya, P.: On the relation between the ideal cipher and the random oracle models. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 184–206. Springer, Heidelberg (2006). https://doi.org/10.1007/11681878_10
34. Dodis, Y., Puniya, P.: Feistel networks made public, and applications. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 534–554. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72540-4_31
35. Dziembowski, S., Maurer, U.M.: Optimal randomizer efficiency in the bounded-storage model. J. Cryptol. **17**(1), 5–26 (2004)
36. Fischlin, M., Janson, C., Mazaheri, S.: Backdoored hash functions: immunizing HMAC and HKDF. Cryptology ePrint Archive, Report 2018/362 (2018). http://eprint.iacr.org/2018/362
37. Katz, J., Lucks, S., Thiruvengadam, A.: Hash functions from defective ideal ciphers. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048, pp. 273–290. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16715-2_15

38. Kawachi, A., Numayama, A., Tanaka, K., Xagawa, K.: Security of encryption schemes in weakened random oracle models. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 403–419. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13013-7_24
39. Kiltz, E., O'Neill, A., Smith, A.: Instantiability of RSA-OAEP under Chosen-plaintext attack. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 295–313. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_16
40. Liskov, M.: Constructing an ideal hash function from weak ideal compression functions. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 358–375. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74462-7_25
41. Maurer, U., Renner, R., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24638-1_2
42. Menn, J.: Exclusive: secret contract tied NSA and security industry pioneer. Reuters, December 2013
43. Mironov, I., Stephens-Davidowitz, N.: Cryptographic reverse firewalls. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part III. LNCS, vol. 9057, pp. 657–686. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_22
44. Myers, S.: Efficient amplification of the security of weak pseudo-random function generators. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 358–372. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_22
45. Numayama, A., Isshiki, T., Tanaka, K.: Security of digital signature schemes in weakened random oracle models. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 268–287. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78440-1_16
46. Perlroth, N., Larson, J., Shane, S.: N.S.A. able to foil basic safeguards of privacy on web. The New York Times (2013). http://www.nytimes.com/2013/09/06/us/nsa-foils-much-internet-encryption.html
47. Ristenpart, T., Shacham, H., Shrimpton, T.: Careful with composition: limitations of the indifferentiability framework. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 487–506. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_27
48. Rubinfeld, R.A.: A mathematical theory of self-checking, self-testing and self-correcting programs. Ph.D. thesis, University of California at Berkeley, Berkeley, CA, USA (1991). UMI Order No. GAX91-26752
49. Russell, A., Tang, Q., Yung, M., Zhou, H.-S.: Cliptography: clipping the power of kleptographic attacks. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 34–64. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53890-6_2
50. Russell, A., Tang, Q., Yung, M., Zhou, H.-S.: Generic semantic security against a kleptographic adversary. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 17, pp. 907–922. ACM Press, October 2017
51. Coretti, S., Dodis, Y., Guo, S., Steinberger, J.: Random oracles and non-uniformity. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10820, pp. 227–258. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_9
52. Schneier, B., Fredrikson, M., Kohno, T., Ristenpart, T.: Surreptitiously weakening cryptographic systems. Cryptology ePrint Archive, Report 2015/097 (2015). http://eprint.iacr.org/2015/097

53. Soni, P., Tessaro, S.: Public-seed pseudorandom permutations. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part II. LNCS, vol. 10211, pp. 412–441. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56614-6_14
54. Young, A., Yung, M.: The dark side of "black-box" cryptography, or: should we trust capstone? In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 89–103. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68697-5_8
55. Young, A., Yung, M.: Kleptography: using cryptography against cryptography. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 62–74. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_6