The Random Oracle Methodology, Revisited*

Ran Canetti† Oded Goldreich‡ Shai Halevi§

August 6, 2002

Abstract

We take a critical look at the relationship between the security of cryptographic schemes in the Random Oracle Model, and the security of the schemes that result from implementing the random oracle by so called "cryptographic hash functions".

The main result of this paper is a negative one: There exist signature and encryption schemes that are secure in the Random Oracle Model, but for which *any implementation* of the random oracle results in insecure schemes.

In the process of devising the above schemes, we consider possible definitions for the notion of a "good implementation" of a random oracle, pointing out limitations and challenges.

Keywords: Correlation Intractability,

- Cryptography (Encryption and Signature Schemes, The Random Oracle model);
- Complexity Theory (diagonalization, application of CS-Proofs).

*Extended abstract has appeared in the *Proc. of the 30th ACM Symp. on Theory of Computing (STOC)*, pages 209–218, 1998.

†IBM Watson, P.O. Box 704, Yorktown Height, NY 10598, USA. E-mail: `canetti@watson.ibm.com`

‡Department of Computer Science, Weizmann Institute of Science, Rehovot, ISRAEL. E-mail: `oded@wisdom.weizmann.ac.il`. Work done while visiting LCS, MIT. Partially supported by DARPA grant DABT63-96-C-0018.

§IBM Watson, P.O. Box 704, Yorktown Height, NY 10598, USA. E-mail: `shaih@watson.ibm.com`

Contents

1 Introduction

A popular methodology for designing cryptographic protocols consists of the following two steps. One first designs an *ideal* system in which all parties (including the adversary) have oracle access to a truly random function, and proves the security of this ideal system. Next, one replaces the random oracle by a "good cryptographic hashing function" (such as MD5 or SHA), providing all parties (including the adversary) with a succinct description of this function. Thus, one obtains an *implementation* of the ideal system in a "real-world" where random oracles do not exist. This methodology, explicitly formulated by Bellare and Rogaway [5] and hereafter referred to as the *random oracle methodology*, has been used in many works (see, for example, [14, 35, 23, 32, 5, 27, 6, 33]).

Although the random oracle methodology seems to be useful in practice, it is unclear how to put this methodology on firm grounds. One can indeed make clear statements regarding the security of the ideal system, but it is not clear what happens when one replaces the random oracle by a "fully specified implementation". What one would have liked is a realizable construct that, when used to replace the random oracle, maintains the security of the ideal scheme. The purpose of this work is to point out fundamental difficulties in proceeding towards this goal.

We demonstrate that the traditional approach of providing a single robust definition that supports a wide range of applications is bound to fail. That is, one cannot expect to see definitions such as of pseudorandom generators or functions [7, 36, 19], and general results of the type saying that these can be used in any application in which parties are restricted merely by computing resources. Specifically, we identify a specific property of the random oracle, that seems to capture one aspect of the random oracle methodology (and in particular seems to underline heuristics such as the Fiat–Shamir transformation of a three-round identification scheme into a signature scheme in the [14]). We show that even a minimalistic formulation of this property, called correlation intractability, cannot be obtained by any "fully specified implementation".

To demonstrate the implications of the above to the security of cryptographic systems, we show that systems whose security relies on the "correlation intractability" of their oracle may be secure in the Random Oracle Model, and yet be insecure when implemented using any fully specified function (or function ensemble). In particular, we describe schemes for digital signatures and public-key encryption that are secure in the Random Oracle Model, but for which any implementation yields insecure schemes.

1.1 The Setting

For the purpose of the following discussion, a cryptographic system consists of a set of parties, which are modeled by probabilistic polynomial time interactive Turing machines. A cryptographic application comes with a security requirement specifying the adversary's abilities and when the latter is considered successful. The abilities of the adversary include its computational power (typically, an arbitrary polynomial-time machine) and the ways in which it can interact with the other parties. The *success* of the adversary is defined by means of a predetermined polynomial-time predicate of the application's global view. (The application's global view consists of the initial inputs of all the parties and of the adversary, their internal coin tosses, and all the messages that were exchanged among them.) A system is considered secure if any adversary with the given abilities has only a negligible probability of success (or, in some cases, only a negligible advantage over a "trivial attack").

1.1.1 The Random Oracle Model

In a scheme that operates in the Random Oracle Model, all parties (including the adversary) interact with one another as usual interactive machines, but in addition they can make oracle queries. It is postulated that all oracle queries, regardless of the identity of the party making them, are answered by a single function, denoted $\mathcal{O}$, that is uniformly selected among all possible functions. The set of possible functions is determined by a length function, $\ell_{\text{out}}(\cdot)$, and by the security parameter of the system. Specifically, given security parameter k we consider functions mapping $\{0,1\}^*$ to $\{0,1\}^{\ell_{\text{out}}(k)}$. A set of interactive oracle machines as above corresponds to an *ideal system for one specific application.* Security of an ideal system is defined as usual. That is, an ideal system is considered secure if any adversary with the given abilities (including oracle access) has only a negligible probability of success (or only a negligible advantage). Here the probability is taken also over the choices of the random oracle.

1.1.2 Implementing an ideal system

Since most real-world systems do not have access to a random oracle, there is a need to "implement" the random oracle aspect of the ideal systems from above. The soundness of the random oracle methodology depends on finding a suitable notion of implementation, such that whenever the ideal system is secure in the Random Oracle Model, the implementation will be secure in the standard model. Furthermore, the implementation should be directly available (i.e., fully specified) to each party.[1] However, all the notions that we consider in this work fail poorly at this challenge.

Loosely speaking, by "implementing" a particular ideal system we mean using an easy-to-evaluate function f instead of the random oracle. That is, whenever the ideal system queries the oracle with a value x, the implementation instead evaluates $f(x)$. In this work, we examine three formalizations of this notion. First we briefly examine (and discard of) the notion of implementation by a single function. Then we discuss implementation by a function ensemble, which is the notion we use through most of the paper. Finally, we discuss a more stringent notion, where the functions in the ensemble can only be evaluated on inputs of a pre-determined (short) length.

Implementation by a single function. This is perhaps the most "natural" notion, in that it corresponds to the common practice of using a fixed function (e.g., SHA-1) to replace the oracle. Here, an ideal system (for some specific application), Π, is transformed into a real system (for the same application) by transforming each interactive oracle machine, into a standard interactive machine in the natural manner. That is, each oracle call is replaced by the evaluation of a fixed function f on the corresponding query.[2]

The above system is called an *implementation of* Π *using function* f. The adversary, attacking this implementation, may mimic the behavior of the adversary of the ideal system, by evaluating f at arguments of its choice, but it may also do other things. In particular, it may obtain some global insight into the structure of the function f, and use this insight towards its vicious goals. An implementation is called secure if any adversary attacking it may succeed only with negligible

[1] One implementation that is clearly sound, is to replace the random function by a pseudorandom one, whose seed remains secret. (Presumably, for this to work there should be an online trusted party who knows the seed and can evaluate the function.) However, this implementation is not fully specified (i.e., it is not directly available to the users). We stress that the random oracle methodology is typically applied in settings where we need a fully specified implementation that the parties can evaluate on their own.

[2] Formally, the function f also takes as input the security parameter k, so that the function $f(k,\cdot)$ maps $\{0,1\}^*$ to $\{0,1\}^{\ell_{\text{out}}(k)}$.

probability, where the success event is defined exactly as in the ideal system (i.e., it is defined by the same polynomial-time computable predicate of the application's global view).

Using this notion of an implementation, we would like to say that a function f is a "good implementation of a random oracle" if for any ideal system Π, security of Π implies security of the implementation of Π using f. It is very easy to see, however, that no (single) polynomial-time computable function can provide a good implementation of a random oracle. Consider, for example, a candidate function f. Then, a (contrived) application for which f does not provide a good implementation consists of an oracle machine (representing an honest party) that upon receiving a message m, makes query m to the oracle and reveals its private input if the oracle answers with $f(m)$. Suppose that the adversary is deemed successful whenever the honest party reveals its private input. Clearly, this ideal system is secure (in the Random Oracle Model), since the random oracle will return the value $f(m)$ only with negligible probability; however, its implementation using f is certainly not secure.

One should not be surprised by the failure of the single-function notion of implementation. Indeed, this notion fails even to be collision-intractable (e.g., it definitely fails with respect to non-uniform polynomial-size circuits), whereas a random oracle is definitely collision-intractable, even w.r.t non-uniform polynomial-size circuits. Indeed, a collision-intractable function is typically modeled not as a single function, but rather as a collection (or ensemble) of functions, where a function is chosen at random from the ensemble and made public once and for all. We thus turn our attention to possible corresponding implementations of the random oracle by function ensembles.

Implementation by a function ensemble. In this setting, we have a "system set-up" phase, in which the function is selected once and for all, and its description is available to all parties.[3] After the set-up phase, this function is used in place of the random oracle just as above. A little more precisely, we consider a function ensemble $\mathcal{F} = \{F_k | k \in \mathsf{N}\}$, where

$$F_k = \{f_s : \{0,1\}^* \to \{0,1\}^{\ell_{\mathrm{out}}(k)}\}_{s \in \{0,1\}^k} , \tag{1}$$

such that there exists a polynomial time algorithm that, on input s and x, returns $f_s(x)$. Just like the random oracle, the ensemble's functions are defined for any input length, although any user and (feasible) adversary will only invoke them on inputs of length bounded by a polynomial in their description length, $|s|$. (Indeed, protocols in the random oracle model often assume that the random oracle is defined for all input lengths.) The implementation of an ideal system, Π, by the function ensemble $\mathcal{F}$ is obtained as follows. On security parameter k, we uniformly select $s \in \{0,1\}^k$, and make s available to all parties including the adversary. Given this initialization phase, we replace each oracle call of an interactive oracle machine by the evaluation of the function f_s on the corresponding query. The resulting system is called an *implementation of* Π *using function ensemble* $\mathcal{F}$.

Again, the adversary may mimic the behavior of the adversary in the Random Oracle Model by evaluating f_s at arguments of its choice, but it can also use its knowledge of the description of f_s in any arbitrary way. Such a real system is called secure if any adversary attacking it has only a negligible probability of success, where the probability is taken over the random choice of s as well as the coins of all the parties. As before, we would like to say that an ensemble $\mathcal{F}$ provides a "good implementation of a random oracle" if for every ideal system Π, if Π is secure then so

[3] In this work we consider examples of public key signature and encryption schemes, where the set-up step is combined with the key-generation step of the original scheme.

is the implementation of Π using $\mathcal{F}$. Notice that in this case, the contrived example from above does not work anymore, since the success event must be independent of the random choice of s. Nonetheless, this work implies that no function ensemble can provide a good implementation of a random oracle.

Restricted function ensembles and other notions. Although the above notions seem like the most "natural" ways of defining an implementation of the random oracle (and they correspond to the common practice of using a so called "cryptographic hash function" to replace the oracle), there still may be other interesting notions. One such example is the notion of function ensembles that are defined over finite domains. That is, instead of considering functions of the form $f_s : \{0,1\}^* \to \{0,1\}^{\ell_{\text{out}}(|s|)}$, one may consider functions of the form $f_s : \{0,1\}^{\ell_{\text{in}}(|s|)} \to \{0,1\}^{\ell_{\text{out}}(|s|)}$. Furthermore, the function description (i.e., s) may be longer than the input and output lengths (i.e., $\ell_{\text{in}}(|s|)$ and $\ell_{\text{out}}(|s|)$). Note that syntactically, such function ensembles can only "implement" a similarly-restricted random oracle (i.e., $\mathcal{O}: \{0,1\}^{\ell_{\text{in}}(k)} \to \{0,1\}^{\ell_{\text{out}}(k)}$). Furthermore, most of our negative results hold also with respect to such restricted "implementations" (see Section 5).

1.2 Our Results

The main results in this paper refer to the notion of implementing a variable input-length oracle by a function ensemble (of functions with variable input-length as in Eq. (1)). Thus, unless we explicitly say otherwise, when we talk about implementing the Random Oracle Model by function ensembles we refer to this notion.

1.2.1 Correlation intractability

One property we certainly expect from a good implementation of a random oracle, is that it should be infeasible to find inputs to the function that stand in some "rare" relationship with the corresponding outputs. Indeed, many applications of the random-oracle methodology (such as the Fiat-Shamir heuristic) assume that it is infeasible to find input-output pairs that stand in a particular relations induced by the application. Trying to formulate this property, we may require that given the description of the function, it is hard to find a sequence of pre-images that together with their images (under this function) satisfy some given relation. Clearly, this can only hold for relations for which finding such sequences is hard in the Random Oracle Model. That is, IF it is hard to find a sequence of pre-images that together with their images under a random oracle satisfy relation R, THEN given the description of a "good" function f_s it should be hard to find a sequence of pre-images that together with their images under f_s satisfy R.

In most of this work we mainly consider the task of finding a *single* pre-image that together with its image satisfies some property. (The case of relations with larger arity is discussed in Section 5, in connection with restricted ensembles.) Loosely speaking, a binary relation is called evasive if when given access to a random oracle $\mathcal{O}$, it is infeasible to find a string x so that the pair $(x, \mathcal{O}(x))$ is in the relation. (For instance, the relation $\{(x, 0^{\ell_{\text{out}}(k)}) : x \in \{0,1\}^*\}$ is evasive. The relation $\{(x, 0y) : x \in \{0,1\}^*, y \in \{0,1\}^{\ell_{\text{out}}(k)-1}\}$ is not.) A function ensemble $\mathcal{F}$ is called correlation intractable if for every evasive binary relation, given the description of a uniformly selected function $f_s \in F_k$ it is infeasible to find an x such that $(x, f_s(x))$ is in the relation.[4] We show that

[4] The more general notion is that of correlation intractability with respect to multiple input-output pairs. The above notion that only talks about one pair should really be called "1-input" correlation intractability. Still in this paper we omit the 1-input qualifiers for ease of presentation. The fact that the following (negative) result refers even to 1-input correlation intractability only makes it stronger.

Informal Theorem 1.1 *There exist no correlation intractable function ensembles.*

Restricted correlation intractability. The proof of the above negative result relies on the fact that the description of the function is shorter than the input used in the attack. Thus, we also investigate (in Section 5) the case where one restricts the function f_s to inputs whose length is less than the length of s. We show that the negative result can be extended to the case where the function description is shorter than the sum of the lengths of the input and output of the function. Furthermore, when one considers the notion of correlation intractability for relations on sequences of inputs and outputs, then the negative result holds as long as the total length of all the inputs and outputs is more than the length of the function description.

Our results still leave open the possibility that there exist function ensembles that are correlation intractable with respect to input-output sequences of total *a-priori bounded* length. However, such ensembles may be useful only in applications where the number of invocations of the cryptosystem is a-priori bounded (or where the security of the system depends only on an a-priori bounded partial history of invocations).[5]

1.2.2 Failures of the Random Oracle Methodology

Upon formulating the random oracle methodology, Bellare and Rogaway did warn that a proof of security in the Random Oracle Model should not be taken as guarantee to the security of implementations (in which the Random Oracle is replaced by functions such as MD5) [5]. However, it was widely believed that a security proof in the Random Oracle Model means that there are no "structural flaws" in the scheme. That is, it was believed that any attack against an implementation of this scheme must take advantage of some "specific flaws in the implementation". A related common belief was that a proof of security in the Random Oracle Model precludes "generic attacks" that work for any implementation. In this work we demonstrate that these beliefs were unfounded. Specifically, we show that

Informal Theorem 1.2 *There exists encryption and signature schemes that are secure in the Random Oracle Model, but have* NO SECURE IMPLEMENTATION *by function ensembles. Moreover, each of these schemes has a "generic adversary", that when given as input the description of an implementation of the oracle, breaks the scheme that uses this implementation.*

The encryption and signature schemes presented to prove Theorem 1.2 are "unnatural". We do not suggest that a statement as above holds with respect to schemes presented in the literature. Still, the lesson is that the mere fact that a scheme is secure in the Random Oracle Model does not necessarily imply that a particular implementation of it (in the real world) is secure, or even that this ideal scheme has *any secure implementation at all.* In fact, our techniques are quite general and can be applied to practically *any* cryptographic application. That is, given an ideal cryptographic application A, we can construct an ideal cryptographic application A' such that A' is just as secure as A (in the Random Oracle Model), but A' has *no secure implementation.*

An afterthought. Trying to explain our negative results, we note that the beliefs reviewed before Theorem 1.2 seem to assume that the only "reasonable thing" that a generic attack can do with a description of the function implementing the oracle, is to invoke it on inputs of its choice. This

[5] We note that the Fiat-Shamir heuristic for transforming interactive identification protocols into signature schemes [14] does not fall into the above category, since the function's seed needs to be fixed with the public key, and used for signing polynomially many messages, where the polynomial is not a-priori known.

oversight, which can be traced to the conjectured difficulty of "reverse engineering", ignores the computational theoretic fact that a code of a program (or part of it) can be fed to the program itself resulting in "unexpected" behavior. Indeed, this is essentially what our "generic attacker" does. In retrospect, several subsequent works (e.g., [3, 1, 2]) demonstrated that having a description of a function is much more powerful than just having a black-box access to that function.

1.3 Techniques

Our proof of Theorem 1.2 uses in an essential way non-interactive CS-proofs (in the Random Oracle Model), as defined and constructed by Micali [27].[6] Interestingly, we only use the fact that non-interactive CS-proofs exist in the Random Oracle Model, and do not care whether or not these ideal CS-proofs have an implementation using function ensembles (nor if non-interactive CS-proofs exists at all outside of the Random Oracle Model). Specifically, CS-proofs are used to "effectively verify" *any* polynomial-time verifiable statement within time that is bounded by one *fixed* polynomial. Furthermore, we use the fact that the definition of CS-proofs guarantees that the complexity of generating such proofs is polynomial in the time required for ordinary verification. See further discussion in Section 2.2.

1.4 Related Work

1.4.1 Previous Work

Correlation intractability. Our definition of correlation-intractability is related to a definition by Okamoto [32]. Using our terminology, Okamoto considers function ensembles for which it is infeasible to form input-output relations with respect to a specific evasive relation [32, Def. 19] (rather than all such relations). He uses the assumption that such function ensembles exists, for a specific evasive relation in [32, Thm. 20].

Special-purpose properties of the Random Oracle Model. First steps in the direction of identifying and studying useful special-purpose properties of the Random Oracle Model have been taken by Canetti [8]. Specifically, Canetti considered a property called "perfect one-wayness", provided a definition of this property, constructions that possess this property (under some reasonable assumptions), and applications for which such functions suffice. Additional constructions have been suggested by Canetti, Micciancio and Reingold [11]. Another context where specific properties of the random oracle where captured and realized is the signature scheme of Gennaro, Halevi and Rabin [15].

1.4.2 Subsequent Work

All works surveyed in this subsection have appeared following the preliminary version of the current work [10].

Relation to Zero-Knowledge proofs. Hada and Tanaka observed that the existence of even restricted correlation intractable functions (in the non uniform model) would be enough to prove that 3-round auxiliary-input zero-knowledge AM proof systems only exist for languages in BPP [24]. (Recall that auxiliary-input zero-knowledge is seemingly weaker than black-box zero-knowledge, and

[6] The underlying ideas of Micali's construction [27] can be traced to Kilian's construction [26] and to the Fiat–Shamir transformation [14] (which is sound in the Random Oracle Model).

so the result of [24] is incomparable to prior work of Goldreich and Krawczyk [20] that showed that constant-round auxiliary-input zero-knowledge AM proof systems only exist for languages in BPP.)

Relation to "magic functions". Following [24], Dwork *et. al.* investigated the notion of "magic functions", which is related to our correlation intractable functions [13]. Like correlation intractability, the definition of "magic functions" is motivated by the quest to capture the properties that are required from the hash function in the Fiat-Shamir heuristic. Correlation intractability seems like a general and natural property, but is not known to be either necessary or sufficient for the Fiat-Shamir heuristic (which is a special case of the random oracle methodology). In contrast, "magic functions" are explicitly defined as "functions that make the Fiat-Shamir heuristic work". In their paper [13], Dwork *et. al.* demonstrated a relation between "magic functions" and 3-round zero-knowledge, similar to the relation between correlation intractability and zero-knowledge exhibited in [24]. Specifically, they showed that the existence of "magic functions" implies the non-existence of some kind of 3-round zero-knowledge proof systems, as well as a weakened version of a converse theorem.

On obfuscating a pseudorandom function ensemble. In their work regarding the impossibility of code obfuscators, Barak *et. al.* [3] have complemented Theorem 1.2 in the following sense. Recall that Theorem 1.2 asserts the existence of (contrived) protocols that are secure in the idealized Random Oracle Model, but have *no secure implementation* by function ensembles. In contrast, the results in [3] imply that a *natural method of obtaining adequate function ensembles* fails to yield secure implementations for *any protocol* that is secure in the random oracle model. Specifically, the method shown to fail is applying any "code obfuscator" (i.e,, a transformation that changes the programs code without changing its functionality) to an ensemble of pseudorandom functions (i.e., an ensemble of functions that cannot be distinguished from a random oracle when only given oracle access to the function [19]).

On the usefulness of the code of a program. In continuation to the afterthought in Section 1.2.2, we mention that the advantage of given a program's code rather than merely oracle access to it has been further demonstrated in subsequent works [3, 1, 2]. In particular, Barak *et. al.* [3] use the code of a program in order to guide a corresponding computation with encrypted intermediate results. Barak [1] (as well as [2]) shows that the code of an adversary can be used to generate certain simulated transcripts that are indistinguishable from the real execution (whereas these specific transcripts cannot be generated while only using oracle access to the adversary's program). Needless to say, none of these work "reverse engineers" the code in any natural sense (that is, there is no attempt to "understand" or "interpret" the code). Rather, they only use the fact that such a short code exists.

On another failure of the Random Oracle Methodology. As stated in Theorem 1.2, there are *specific* schemes that are secure in the Random Oracle Model, and still have no secure implementation by function ensembles. A recent work of Nielsen [30] shows that there are natural cryptographic tasks that can be securely realized in the Random Oracle Model, but cannot be securely realized in the standard model without a random oracle. (The task considered by Nielsen is non-committing encryption [9]). Note that Nielsen's result is more general that Theorem 1.2 in two ways. Firstly, Nielsen refers to a (natural) *task* rather than to a specific protocol that securely implements it in the Random Oracle Model. Secondly, Nielsen rules out *any* implementation of the task in the standard model, rather than only ruling out implementations resulting by replacing oracle calls (to the random oracle) by function evaluations (for a function selected at random in a function ensemble). Analogously, our Theorem 1.1 can be viewed as asserting that there exists a natural *tool* (i.e., correlation intractable functions) that can be securely implemented in the Random Oracle Model but not in the standard model.

1.5 Organization

Section 2 presents syntax necessary for the rest of the paper as well as review the definition of CS-proofs. Section 3 discusses the reasoning that led us to define the correlation intractability property, and prove that even such a minimalistic definition cannot be met by function ensembles. Section 4 presents our main negative results – demonstrating the existence of secure ideal signature and encryption schemes that do not have secure implementations. In Section 5 we extend these negative results (in some cases) to ensembles with length restrictions. Also in that section, we discuss the margins to which we could not extend these negative results, and hint on some other possible directions that one may explore in the face of these negative results. In Section 6 we present three different perspectives on the results in this paper, and discuss some directions for future research.

2 Preliminaries

We consider probability spaces defined over executions of probabilistic machines. Typically, we consider the probability that an output generated by one machine M_1 satisfies a condition that involves the execution of a second machine M_2. For example, we denote by $\Pr[y \leftarrow M_1(x)\,,\,|y| = |x|\;\&\;M_2(y)=1]$ the probability that on input x, machine M_1 outputs a string that has length $|x|$ and is accepted by machine M_2. That is, y in the above notation represents a random variable that may be assigned arbitrary values in $\{0,1\}^*$, conditions are made regarding this y, and we consider the probability that these conditions are satisfied when y is distributed according to $M_1(x)$.

2.1 Function Ensembles

To make the discussion in the Introduction more precise, we explicitly associate a length function, $\ell_{\text{out}} : \mathsf{N} \to \mathsf{N}$, with the output of the random oracle and its candidate implementations. We usually assume that the length functions are super-logarithmic and polynomially bounded (i.e. $\omega(\log k) \leq \ell_{\text{out}}(k) \leq \text{poly}(k)$). We refer to an oracle with length function ℓ_{out} as an ℓ_{out}-oracle. On security parameter k, each answer of the oracle is a string of length $\ell_{\text{out}}(k)$. A candidate implementation of a random ℓ_{out}-oracle is an ℓ_{out}-ensemble as defined below.

Definition 2.1 (function ensembles) *Let $\ell_{\text{out}} : \mathsf{N} \to \mathsf{N}$ be a length function. An ℓ_{out}-ensemble is a sequence $\mathcal{F} = \{F_k\}_{k \in N}$ of families of functions, $F_k = \{f_s : \{0,1\}^* \to \{0,1\}^{\ell_{\text{out}}(k)}\}_{s \in \{0,1\}^k}$, so that the following holds*

Length requirement. *For every $s \in \{0,1\}^k$ and every $x \in \{0,1\}^*$, $|f_s(x)| = \ell_{\text{out}}(k)$.*

Efficiency requirement. *There exists a polynomial-time algorithm* $\textsc{Eval}$ *so that for every $s, x \in \{0,1\}^*$, it holds that* $\textsc{Eval}(s,x) = f_s(x)$.

In the sequel we often call s the description or the seed of the function f_s.

Remark 2.2 The length of the seed in the above definition serves as a "security parameter" and is meant to control the "quality" of the implementation. It is important to note that although $f_s(\cdot)$ is syntactically defined on every input, in a cryptographic applications it is only used on inputs of length at most $\mathrm{poly}(|s|)$. (Typically, the exact polynomial depends on the application in which this function ensemble is used.) We stress that all results presented in this paper refer to such usage.

Remark 2.3 One may even envision applications in which a more stringent condition on the use of f_s holds. Specifically, one may require that the function f_s be only applied to inputs of length at most $\ell_{\mathrm{in}}(|s|)$, where $\ell_{\mathrm{in}} : \mathsf{N} \to \mathsf{N}$ is a specific length function (e.g., $\ell_{\mathrm{in}}(k) = 2k$). We discuss the effects of making such a stringent requirement in Section 5.

2.2 CS Proofs

Our construction of signature and encryption schemes that are secure in the Random Oracle Model but not in the "real world" uses CS-proofs as defined and constructed by Micali [27]. Below, we briefly recall the relevant definitions and results.

A CS-proof system consists of a prover, PRV, that is trying to convince a verifier, VER, of the validity of an assertion of the type *machine M accepts input x within t steps.*[7] A central feature of CS-proofs is that the running-time of the prover on input x is (polynomially) related to the *actual* running time of $M(x)$ rather than to the global upper bound t; furthermore, the verifier's running-time is poly-logarithmic related to t. (These conditions are expressed in the *additional efficiency requirements* in Definition 2.4 below.)

In our context, we use non-interactive CS-proofs that work in the Random Oracle Model; that is, both prover and verifier have access to a common random oracle. The prover generates an alleged proof that is examined by the verifier. A construction for such CS-proofs was presented by Micali [27], using ideas that can be traced to Kilian's construction [26], and requires no computational assumptions. Following is the formulation of CS-proofs, as defined by Micali.

In the formulation below, the security parameter k is presented in unary to both parties, whereas the global time bound t is presented in unary to the prover and in binary to the verifier. This allows the (polynomial-time) prover to run in time polynomial in t, whereas the (polynomial-time) verifier may only run in time that is poly-logarithmic in t. (Observe that it is *not required* that t is bounded above by a polynomial in $|x|$. In fact, in our arguments, we shall use a slightly super-polynomial function t (i.e., $t(n) = n^{\log n}$).) Finally, we mention that both the prover and the verifier in the definition below are required to be deterministic machines. See some discussion in Remark 2.6 below.

Definition 2.4 (Non-interactive CS proofs in the Random Oracle Model) *A CS-proof system consists of two* (deterministic) *polynomial-time oracle machines, a prover* PRV *and a verifier* VER*, operating as follows:*

- *On input $(1^k, \langle M\rangle, x, 1^t)$ and access to an oracle $\mathcal{O}$, the prover computes a proof $\pi = \mathrm{PRV}^{\mathcal{O}}(1^k, \langle M\rangle, x, 1^t)$ such that $|\pi| \leq \mathrm{poly}(k, |\langle M\rangle|, |x|, \log t)$.*
- *On input $(1^k, \langle M\rangle, x, t, \pi)$, with t encoded in binary, and access to $\mathcal{O}$, the verifier decides whether to accept or reject the proof π* (i.e., $\mathrm{VER}^{\mathcal{O}}(1^k, \langle M\rangle, x, t, \pi) \in \{\texttt{accept}, \texttt{reject}\}$).

[7] When t is presented in binary, such valid assertions form a complete language for the class (deterministic) exponential time.

The proof system satisfies the following conditions, where the probabilities are taken over the random choice of the oracle $\mathcal{O}$:

Perfect completeness: *For any M, x, t such that machine M accepts the string x within t steps, and for any k,*

$$\Pr_{\mathcal{O}}\left[\begin{array}{l}\pi \leftarrow \mathrm{PRV}^{\mathcal{O}}(1^k, \langle M\rangle, x, 1^t), \\ \mathrm{VER}^{\mathcal{O}}(1^k, \langle M\rangle, x, t, \pi) = \texttt{accept}\end{array}\right] = 1$$

Computational soundness: *For any polynomial time oracle machine* BAD *and any input $w = (\langle M\rangle, x, 1^t)$ such that M* does not *accepts x within t steps, it holds that*

$$\Pr_{\mathcal{O}}\left[\begin{array}{l}\pi \leftarrow \mathrm{BAD}^{\mathcal{O}}(1^k, \langle M\rangle, x, 1^t), \\ \mathrm{VER}^{\mathcal{O}}(1^k, \langle M\rangle, x, t, \pi) = \texttt{accept}\end{array}\right] \leq \frac{poly(k + |w|)}{2^k}$$

Additional efficiency conditions:[8] *The running-time of the prover* PRV *on input $(1^k, \langle M\rangle, x, 1^t)$ is (polynomially) related to the* actual *running time of $M(x)$, rather than to the global upper bound t. That is, there exists a fixed polynomial $p(\cdot)$, such that*

$$T_{PRV}\left(1^k, \langle M\rangle, x, 1^t\right) \leq p(k, \min\{t, T_M(x)\})$$

where $T_A(x)$ denotes the running time of machine A on input x.

Remark 2.5 (Oracle output length) The above definition does not specify the output length of the oracle (i.e., the length of the answers to the oracle queries). In some cases it is convenient to identify this output length with the security parameter, but in many case we do not follow this convention (e.g., in Proposition 2.8 below). In any case, it is trivial to implement an oracle with one output length given an oracle with different output length, so we allow ourselves to ignore this issue.

Remark 2.6 (Deterministic verifier) Recall that Definition 2.4 mandates that both the prover and verifier are deterministic. Indeed this deviates from the tradition (in this area) of allowing the verifier to be probabilistic; but Micali's construction (in the Random Oracle Model) happens to employ a deterministic verifier (cf. [27]). This issue is not essential to our main results, but plays an important role in the proof of Proposition 5.8 (due to K. Nissim). We note that when working in the Random Oracle Model (and only caring about completeness and soundness), one may assume without loss of generality that the prover is deterministic (because it can obtain adequate randomness by querying the oracle). This does not hold with respect to the verifier, since its coin tosses may need to be unknown to the prover.

Theorem 2.7 (Micali [27]) *There exists a non-interactive CS proof system in the Random Oracle Model.*

For the proof of our construction (Theorem 4.4), we need a different soundness condition than the one from above. Specifically, we need to make sure that given the machine M (and the

[8] By the above, the running time of PRV on input $(1^k, \langle M\rangle, x, 1^t)$ is at most $\mathrm{poly}(k, |\langle M\rangle|, |x|, t)$, whereas the running time of VER on input $(1^k, \langle M\rangle, x, t, \pi)$ is at most $\mathrm{poly}(k, |\langle M\rangle|, |x|, |\pi|, \log t)$. The additional efficiency condition provides even lower running time bound for the prover. Note that if M runs for much less time than t, the prover may not even have enough time to read its entire input.

complexity bound t), it is hard to find *any pair* (x, π) such that M does not accept x within t steps and yet VER will accept π as a valid CS-proof to the contrary. One way to obtain this soundness property from the original one, is by postulating that when the verifier is given a proof for an assertion $w = (\langle M \rangle, x, t)$, it uses security parameter $k + |w|$ (rather than just k). Using a straightforward counting argument we get:

Proposition 2.8 *Let* (PRV, VER) *be a CS proof system. Then for every polynomial time oracle machine* BAD*, there exists a polynomial* $q(\cdot)$*, such that for every* k *it holds that*

$$\epsilon_{\text{bad}}(k) \stackrel{\text{def}}{=} \Pr_{\mathcal{O}}\left[\begin{array}{c} (w,\pi) \leftarrow \text{BAD}^{\mathcal{O}}(1^k), \ \textit{where } w = (\langle M \rangle, x, t), \\ \textit{s.t. machine } M \textit{ does not accept } x \textit{ within } t \textit{ steps} \\ \textit{and yet } \text{VER}^{\mathcal{O}}(1^{k+|w|}, w, \pi) = \texttt{accept} \end{array}\right] \leq \frac{q(k)}{2^k}$$

3 Correlation Intractability

In this section we present and discuss the difficulty of defining the intuitive requirement that a function ensemble "behaves like a random oracle" even when its description is given. We first comment that an "obvious over-reaching definition", which amount to adopting the pseudorandom requirement of [19], fails poorly. That is, we cannot require that an (efficient) algorithm that is given the description of the function cannot distinguish its input-output behavior from the one of a random function, because the function description determines its input-output behavior.

Towards a definition. Although we cannot require the value of a fully specified function to be "random", we may still be able to require that it has some "unpredictability properties". For example, we may require that, given a description of a family and a function chosen at random from a this family, it is hard to find two pre-images that the function maps to the same image. Indeed, this sound definition coincides with the well-known *collision-intractability* property [12]. Trying to generalize, we may replace the "equality of images" relation by any other relation among the pre-images and images of the function. Namely, we would like to say that an ensemble is *correlation intractable* if for *any* relation, given the description of a randomly chosen function, it is infeasible to find a sequence of pre-images that together with their images satisfy this relation.

This requirement, however, is still unreasonably strong since there are relations that are easy to satisfy even in the Random Oracle Model. We therefore restrict the above infeasibility requirement by saying that it holds only with respect to relations that are hard to satisfy in the Random Oracle Model. That is, IF it is hard to find a sequence of pre-images that together with their images under a random function satisfy relation R, THEN given the description of a randomly chosen function f_s it should be hard to find a sequence of pre-images that together with their images under f_s satisfy R.

This seems to be a minimalistic notion of correlation intractable ensemble of functions, yet we show below that no ensemble can satisfy it. In fact, in the definition below we only consider the task of finding a single pre-image that together with its image satisfies some property. Namely, instead of considering all possible relations, we only consider binary ones. Since we are showing impossibility result, this syntactic restriction only strengthens the result. (When we consider restricted ensembles in Section 5, we will revisit the case of relations with larger arity.)

3.1 Actual Definitions

We start with a formal definition of a relation that is hard to satisfy in the random oracle model.

Definition 3.1 (Evasive Binary Relations) *A binary relation R is said to be evasive with respect to length function $\ell_{\rm out}$ if for any probabilistic polynomial time oracle machine M there is a negligible function*[9] negl *such that*

$$\Pr_{\mathcal{O}}[x \leftarrow M^{\mathcal{O}}(1^k), \text{ and } (x, \mathcal{O}(x)) \in R] \;=\; \text{negl}(k)$$

where $\mathcal{O} : \{0,1\}^* \to \{0,1\}^{\ell_{\rm out}(k)}$ is a uniformly chosen function.

A special case of evasive relations consists of R's for which there exists a negligible function negl$(\cdot)$ so that for all k

$$\sup_{x\in\{0,1\}^*} \left\{ \Pr_{y\in\{0,1\}^{\ell_{\rm out}(k)}}[(x,y)\in R\,] \right\} \;=\; \text{negl}(k)$$

(All the binary relations used in the sequel falls into this category.) The reason such an R is evasive is that any oracle machine, M, making at most poly(k) queries to a random $\mathcal{O}$ satisfies

$$\begin{aligned}\Pr_{\mathcal{O}}[x \leftarrow M^{\mathcal{O}}(1^k),\ (x, \mathcal{O}(x)) \in R] \;&\leq\; \text{poly}(k) \cdot \sup_{x\in\{0,1\}^*} \{ \Pr_{\mathcal{O}}[(x, \mathcal{O}(x)) \in R] \} \\ &\leq\; \text{poly}(k) \cdot \text{negl}(k)\end{aligned}$$

We are now ready to state our minimalistic definition of a correlation intractable ensemble:

Definition 3.2 (correlation intractability) *Let $\ell_{\rm out} : N \to N$ be length function, and let $\mathcal{F}$ be an $\ell_{\rm out}$-ensemble.*

- *Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be a binary relation. We say that $\mathcal{F}$ is correlation intractable with respect to R if for every probabilistic polynomial-time machine M there is a negligible function* negl *such that*

  $$\Pr_{s\in\{0,1\}^k}[x \leftarrow M(s),\ \ (x, f_s(x)) \in R] = \text{negl}(k)$$

  where the probability is taken over the choice of $s \in \{0,1\}^k$ and the coins of M.

- *We say that $\mathcal{F}$ is correlation intractable, if it is correlation intractable with respect to every evasive binary relation (w.r.t. $\ell_{\rm out}$).*

Remark 3.3 In the above definition we quantify over all evasive binary relations. A weaker notion, called weak correlation intractability, is obtained by quantifying only over all polynomial-time recognizable evasive binary relations (i.e., we only consider those relations R such that there exists a polynomial time algorithm that, given (x, y), decides whether or not $(x, y) \in R$). In the sequel we consider both notions.

3.2 Correlation-intractable ensembles do not exist

Theorem 3.4 *There exist no correlation intractable ensembles, not even in the weak sense.*

Proof: Let $\ell_{\rm out}$ be a length function and let $\mathcal{F} = \{f_s\}$ be an $\ell_{\rm out}$-ensemble. We define the binary relation:

$$R^{\mathcal{F}} \stackrel{\rm def}{=} \bigcup_k \left\{ (s, f_s(s)) : s \in \{0,1\}^k \right\} \tag{2}$$

[9] A function $\mu : \mathsf{N} \to R$ is negligible if for every positive polynomial p and all sufficiently large n's, $\mu(n) < 1/p(n)$.

Clearly, this relation is polynomial-time recognizable, since f_s can be computed in polynomial time. Also, the relation is evasive (w.r.t. $\ell_{\rm out}$) since for every $x \in \{0,1\}^*$ there is at most one $y \in \{0,1\}^{\ell_{\rm out}(k)}$ satisfying $(x,y) \in R^{\mathcal{F}}$, [10] and so

$$\Pr_y[(x,y) \in R^{\mathcal{F}}] \leq 2^{-\ell_{\rm out}(k)} = 2^{-\omega(\log k)} = \mathrm{negl}(k)\,.$$

On the other hand, consider the machine I that computes the identity function, $I(x) = x$ for all x. It violates the correlation intractability requirement, since for all k,

$$\Pr_{s\in\{0,1\}^k}[(I(s), f_s(I(s))) \in R^{\mathcal{F}}] \;=\; \Pr_{s\in\{0,1\}^k}[(s, f_s(s)) \in R^{\mathcal{F}}] \;=\; 1\,.$$

In fact, since $R^{\mathcal{F}}$ is polynomial-time recognizable, even the weak correlation intractability of $\mathcal{F}$ is violated. ■

4 Failures of the Random Oracle Methodology

This section demonstrates that the security of a cryptographic scheme in the Random Oracle Model does not always imply its security under some specific choice of a "good hash function" that is used to implement the random oracle. To prove this statement we construct signature and encryption schemes that are secure in the Random Oracle Model, yet for which *any implementation* of the random oracle (by a function ensemble) yields insecure schemes. Put in other words, although the ideal scheme is secure, any implementation of it is necessarily insecure.

The underlying idea is to start with a secure scheme (which may or may not use a random oracle) and modify it to get a scheme that is secure in the Random Oracle Model, but such that its security is easily violated when trying to replace the random oracle by any ensemble. This is done by using evasive relations as constructed in Theorem 3.4. The modified scheme starts by trying to find a pre-image that together with its image yields a pair in the evasive relation. In case the attempt succeeds, the scheme does something that is clearly insecure (e.g., output the secret key). Otherwise, the scheme behaves as the original (secure) scheme does. The former case (i.e., finding a pair in the relation) will occur rarely in the Random Oracle Model, thus the scheme will maintain its security there. However, it will be easy for an adversary to make sure that the former case always occurs under any implementation of the Random Oracle Model, thus no implementation may be secure.[11] We start with the case of a signature scheme, and present the construction in three steps.

- In the first step we carry out the above idea in a naive way. This allows us to prove a weaker statement, saying that for any function ensemble $\mathcal{F}$, there exists a signature scheme that is secure in the Random Oracle Model, but is not secure when implemented using $\mathcal{F}$.

  This, by itself, means that one cannot construct a function ensemble that provides secure implementation of any cryptographic scheme that is secure in the Random Oracle Model. But it still does not rule out the possibility (ruled out below) that for any cryptographic scheme that is secure in the Random Oracle Model there exists a secure implementation (via a different function ensemble).

[10] Such a y exists if and only if $\ell_{\rm out}(|x|) = \ell_{\rm out}(k)$.

[11] On a higher level, one can think of the attack as trying to "prove to the scheme that its oracle is actually being implemented by an ensemble." If the scheme is convinced, it becomes insecure. Viewed in this light, the use of evasive relations is but one example of how such "proof of implementation" can be constructed.

- In the second step we use diagonalization techniques to reverse the order of quantifiers. Namely, we show that there exists a signature scheme that is secure in the Random Oracle Model, but for which *any* implementation (using any function ensemble) results in an insecure scheme. However, the scheme constructed in this step utilizes signing and verification procedures that run in (slightly) super-polynomial time.

- In the third step we use CS-proofs [27] to get rid of the super-polynomial running-time (of the legitimate procedures), hence obtaining a standard signature scheme that is secure in the Random Oracle Model, but has no secure implementation. Specifically, in this step we use CS-proofs as a tool to "diagonalize against all polynomial-time ensembles in polynomial time". (As noted by Silvio Micali, this technique may be useful also in other settings where diagonalization techniques are applied.)

The reader is referred to [22] for basic terminology regarding signature schemes and corresponding notions of security. As a starting point for our constructions, we use a signature scheme, denoted $\mathcal{S} = (G, S, V)$, where G is the key-generation algorithm, S is the signing algorithm, and V is the verification algorithm. We assume that the scheme (G, S, V) is existentially unforgeable under adaptive chosen message attack, in the Random Oracle Model. We do not need to rely on any computational assumptions here, since one-way functions are sufficient for constructing secure signature schemes [29, 34], and the random oracle can be used to implement one-way functions without any assumptions.[12]

Conventions. In the three steps below we assume, without loss of generality, that the security parameter (i.e., k) is implicit in the keys generated by $G(1^k)$. Also, let us fix some length function $\ell_{\text{out}} : \mathsf{N} \to \mathsf{N}$, which would be implicit in the discussions below (i.e., we assume that the random oracles are all ℓ_{out}-oracles, the relations are evasive w.r.t. ℓ_{out}, etc.).

4.1 First Step

Definition. *Let $\mathcal{S} = (G, S, V)$ be a signature scheme (that may or may not use a random oracle), and let R be any binary relation that is evasive w.r.t. length function ℓ_{out}. Then, by $\mathcal{S}_R = (G, S_R, V_R)$ we denote the following modification of $\mathcal{S}$ that utilizes a random ℓ_{out}-oracle:*

Modified signature, $S_R^{\mathcal{O}}(\mathrm{sk}, \mathrm{msg})$, *of message* msg *using signing key* sk:

1. If $(\mathrm{msg}, \mathcal{O}(\mathrm{msg})) \in R$, output $(\mathrm{sk}, \mathrm{msg})$.
2. Otherwise (i.e., $(\mathrm{msg}, \mathcal{O}(\mathrm{msg})) \notin R$), output $S^{\mathcal{O}}(\mathrm{sk}, \mathrm{msg})$.

Modified verification, $V_R^{\mathcal{O}}(\mathrm{vk}, \mathrm{msg}, \sigma)$, *of alleged signature σ to* msg *using verification key* vk:

1. If $(\mathrm{msg}, \mathcal{O}(\mathrm{msg})) \in R$ then `accept`
2. Otherwise output $V^{\mathcal{O}}(\mathrm{vk}, \mathrm{msg}, \sigma)$.

The key-generation algorithm, G, is the same as in the original scheme $\mathcal{S}$. Item 1 in the signing/verification algorithms is a harmful modification to the original signature scheme. Yet, if R is evasive, then it has little effect on the ideal system, and the behavior of the modified scheme is "indistinguishable" from the original one. In particular,

[12] Alternatively, we could use an 'ordinary' signature scheme, but then our Theorem 4.4 would be conditioned on the existence of one-way functions.

Proposition 4.1 *Suppose that R is evasive* (w.r.t. ℓ_{out}) *and that $\mathcal{S}$ is existentially unforgeable under a chosen message attack in the Random Oracle Model. Then $\mathcal{S}_R$ is also existentially unforgeable under a chosen message attack in the Random Oracle Model.*

Proof: The intuition is that since R is evasive, it is infeasible for the forger to find a message m so that $(m, \mathcal{O}(m)) \in R$. Thus, a forgery of the modified scheme must be due to Item (2), contradicting the security of the original scheme.

Formally, let A_R be an adversary who mounts an adaptive chosen message attack on $\mathcal{S}_R$, and whose success probability in obtaining an existential forgery (in the Random Oracle Model) is $\epsilon_{\text{frg}} = \epsilon_{\text{frg}}(k)$. Assume, toward contradiction, that ϵ_{frg} is not negligible in the security parameter k.

Denote by REL the event in which during an execution of A_R, it hands out a message m for which $(m, \mathcal{O}(m)) \in R$ (either as a query to the signer during the chosen message attack, or as the message for which it found a forgery at the end), and let $\epsilon_{\text{rel}} = \epsilon_{\text{rel}}(k)$ be the probability of that event. Using the hypothesis that R is evasive, we now prove that ϵ_{rel} is negligible in the security parameter k. Suppose, to the contrary, that ϵ_{rel} is not negligible. Then, we can try to efficiently find pairs $(x, \mathcal{O}(x)) \in R$ by choosing a key-pair for $\mathcal{S}$, and then implementing the attack, playing the role of both the signer algorithm and the adversary A_R. With probability ϵ_{rel}, one of A_R's messages during this attack satisfies $(m, \mathcal{O}(m)) \in R$, so just choosing at random one message that was used and outputting it yields a success probability of ϵ_{rel}/q (with q being the number of different messages that are used in the attack). If ϵ_{rel} is not negligible, then neither is ϵ_{rel}/q, contradicting the evasiveness of R.

It is clear that barring the event REL, the execution of A_R against the original scheme $\mathcal{S}$ would be identical to its execution against $\mathcal{S}_R$. Hence the probability that A_R succeeds in obtaining an existential forgery against $\mathcal{S}$ is at least $\epsilon_{\text{frg}} - \epsilon_{\text{rel}}$. Since ϵ_{rel} is negligible, and ϵ_{frg} is not, then A_R's probability of obtaining an existential forgery against $\mathcal{S}$ is also not negligible, contradicting the assumed security of $\mathcal{S}$. ■

The modification to $\mathcal{S}$ enables to break the modified scheme $\mathcal{S}_R$ when implemented with a real ensemble $\mathcal{F}$, in the case where R is the relation $R^{\mathcal{F}}$ from Proposition 3.4. Indeed, as corollary to Propositions 3.4 and 4.1, we immediately obtain:

Corollary 4.2 *For every efficiently computable ℓ_{out}-ensemble $\mathcal{F}$, there exists a signature scheme that is existentially unforgeable under a chosen message attack in the Random Oracle Model, yet when implemented with $\mathcal{F}$, the resulting scheme is totally breakable under an adaptive chosen message attack, and existentially forgeable under a key-only attack.*

Proof: When we use an ensemble $\mathcal{F}$ to implement the random oracle in the scheme $\mathcal{S}_R$, we obtain the following real scheme (which we denote $\mathcal{S}'_R = (G', S'_R, V'_R)$):

$G'(1^k)$**:** Uniformly pick $s \in \{0,1\}^k$, set (sk, vk) $\leftarrow G^{f_s}(1^k)$, and output $(\langle \text{sk}, s \rangle, \langle \text{vk}, s \rangle)$.

$S'_R(\langle \text{sk}, s \rangle, \text{msg})$**:** Output $S_R^{f_s}(\text{sk}, \text{msg})$.

$V'_R(\langle \text{vk}, s \rangle, \text{msg}, \sigma)$**:** Output $V_R^{f_s}(\text{vk}, \text{msg}, \sigma)$.

Consider now what happens when we use the ensemble $\mathcal{F}$ to implement the the scheme $\mathcal{S}_{R^{\mathcal{F}}}$ (recall the definition of $R^{\mathcal{F}}$ from Eq. (2)). Since $R^{\mathcal{F}}$ is evasive, then from Proposition 4.1 we infer that the $\mathcal{S}_{R^{\mathcal{F}}}$ is secure in the Random Oracle Model. However, when we use the ensemble $\mathcal{F}$ to implement the scheme, the seed s becomes part of the public verification-key, and hence is known to the adversary. The adversary can simply output the pair (s,ϵ), that will be accepted by $V'_{R^{\mathcal{F}}}$ as a valid message-signature pair (since $(s, f_s(s)) \in R^{\mathcal{F}}$). Hence, the adversary achieves existential forgery (of $\mathcal{S}'_{R^{\mathcal{F}}}$) under key-only attack. Alternatively, the adversary can ask the legitimate signer for a signature on s, hence obtaining the secret signing-key (i.e., total forgery). ■

4.2 Second Step

Enumeration. For this (and the next) subsection we need an enumeration of all efficiently computable function ensembles. Such enumeration is achieved via an enumeration of all polynomial-time algorithms (i.e., candidates for evaluation algorithms of such ensembles). Several standard technicalities arise. First, enumerating all polynomial-time algorithms is problematic since there is no single polynomial that bounds the running time of all these algorithms. Instead, we fix an arbitrary super-polynomial proper complexity function[13], $t : \mathsf{N} \to \mathsf{N}$ (e.g., $t(n) = n^{\log n}$), and enumerate all algorithms of running-time bounded by t. The latter is done by enumerating all possible algorithms, and modifying each algorithm by adding a time-out mechanism that terminates the execution in case more than $t(|\text{input}|)$ steps are taken. This modification does not effect the polynomial-time algorithms. Also, since we are interested in enumerating ℓ_{out}-ensembles, we modify each function by viewing its seed as a pair $\langle s, x\rangle$ (using some standard parsing rule) and padding or truncating its output to length $\ell_{\text{out}}(|s|)$. Again, this modification has no effect on the ℓ_{out}-ensembles.

Let us denote by $\mathcal{F}^i$ the i^{th} function ensemble according to the above enumeration, and denote by f^i_s the function indexed by s from the ensemble $\mathcal{F}^i$. Below we again use some standard rule for parsing a string α as a pair $\langle i, s\rangle$ and viewing it as a description of the function f^i_s.

Universal ensemble. Let $\mathcal{U} = \{U_k\}_{k\in\mathbf{N}}$ denote the "universal function ensemble" that is induced by the enumeration above, namely $U_k = \{u_{\langle i,s\rangle}\}_{\langle i,s\rangle\in\{0,1\}^k}$ and $u_{\langle i,s\rangle}(x) = f^i_s(x)$. There exists a machine that computes the universal ensemble $\mathcal{U}$ and works in slightly super-polynomial time, t.

Universal relation. Denote by $R^{\mathcal{U}}$ the universal relation that is defined with respect to the universal ensemble $\mathcal{U}$ similarly to the way that $R^{\mathcal{F}}$ is defined with respect to any ensemble $\mathcal{F}$. That is:

$$R^{\mathcal{U}} \stackrel{\text{def}}{=} \bigcup_k \left\{ \Big(\langle i,s\rangle, f^i_s(\langle i,s\rangle)\Big) : \langle i,s\rangle \in \{0,1\}^k \right\}$$

Or, in other words:

$$(x,y) \in R^{\mathcal{U}} \iff \begin{array}{l} y = u_x(x) \\ \text{(i.e., } x = \langle i,s\rangle \text{ and } y = f^i_s(x)) \end{array}$$

Modified signature scheme. Let $\mathcal{S} = (G, S, V)$ be a signature scheme (as above). We then denote by $\mathcal{S}_u = (G, S_u, V_u)$ the modified signature scheme that is derived by using $R^{\mathcal{U}}$ in place of R in the previous construction. Specifically:

$S^{\mathcal{O}}_u(\text{sk}, \text{msg}) \stackrel{\text{def}}{=}$

1. If $(\text{msg}, \mathcal{O}(msg)) \in R^{\mathcal{U}}$ (i.e., if $\text{msg} = \langle i, s\rangle$ and $\mathcal{O}(\text{msg}) = f^i_s(\text{msg})$) then output (sk, msg).
2. Otherwise, output $S^{\mathcal{O}}(\text{sk}, \text{msg})$

[13] Recall that $t(n)$ is a *proper complexity function* (or time-constructible) if there exists a machine that computes $t(n)$ and works in time $O(t(n))$. This technical requirement is needed to ensure that the enumeration itself is computable in time $O(t(n))$.

$V_u^{\mathcal{O}}(\mathrm{vk}, \mathrm{msg}, \sigma) \stackrel{\mathrm{def}}{=}$

1. If $(\mathrm{msg}, \mathcal{O}(msg)) \in R^{\mathcal{U}}$ then `accept`.
2. Otherwise, output $V^{\mathcal{O}}(\mathrm{vk}, \mathrm{msg}, \sigma)$.

We note that since these signature and verification algorithms need to compute $\mathcal{U}$, they both run in time $O(t)$, which is slightly super-polynomial.

Proposition 4.3 *Suppose that $\mathcal{S}$ is existentially unforgeable under a chosen message attack in the Random Oracle Model. Then $\mathcal{S}_u$ is also existentially unforgeable under a chosen message attack in the Random Oracle Model, but implementing it with* any function ensemble *yields a scheme that is totally breakable under chosen message attack and existentially forgeable under key-only attack.*

Proof: Since $R^{\mathcal{U}}$ is evasive, then from Proposition 4.1 it follows that $\mathcal{S}_u$ is secure in the Random Oracle Model. On the other hand, suppose that one tries to replace the random oracle in the scheme by an ensemble $\mathcal{F}^i$ (where i be the index in the enumeration). An adversary, given a seed s of a function in $\mathcal{F}^i$ can then set $\mathrm{msg} = \langle i, s \rangle$ and output the pair (msg, ϵ), which would be accepted as a valid message-signature pair by V_u. Alternatively, it can ask the signer for a signature on this message msg, and so obtain the secret signing-key. ■

4.3 Third step

We now use CS-proofs to construct a new signature scheme that works in the Random Oracle Model. This construction is similar to the one in Subsection 4.2, except that instead of checking that $(\mathrm{msg}, \mathcal{O}(\mathrm{msg})) \in R^{\mathcal{U}}$, the signer/verifier gets a CS-proof of that claim, and it only needs to verify the validity of that proof. Since verifying the validity of a CS-proof can be done much more efficiently than checking the claim "from scratch", the signing and verifications algorithms in the new scheme may work in polynomial time. On the other hand, when the scheme is implemented using the function ensemble $\mathcal{F}^i$, supplying the adequate CS-proof (i.e., for $(\mathrm{msg}, f_s^i(\mathrm{msg})) \in R^{\mathcal{U}}$) only requires polynomial-time (i.e., time polynomial in the time it takes to evaluate f_s^i). This yields the following:

Theorem 4.4 *There exists a signature scheme that is existentially unforgeable under a chosen message attack in the Random Oracle Model, but such that when implemented with any function ensemble, the resulting scheme is existentially forgeable using key-only attack and totally breakable under chosen message attack.*

We note again that unlike the "signature scheme" presented in Subsection 4.2, the signature scheme presented below works in polynomial-time.

Proof: Below we describe such a signature scheme. For this construction we use the following ingredients.

- $\mathcal{S} = (G, S, V)$ is a signature scheme, operating in the Random Oracle Model, that is existentially unforgeable under a chosen message attack.
- A fixed (and easily computable) parsing rule that interpret messages as triples of strings $\mathrm{msg} = \langle i, s, \pi \rangle$.
- The algorithms PRV and VER of a CS-proof system, as described in Section 2.2 above.

- Access to three independent random oracles. This is very easy to achieve given access to one oracle $\mathcal{O}$; specifically, by setting $\mathcal{O}'(x) \stackrel{\text{def}}{=} \mathcal{O}(01x)$, $\mathcal{O}''(x) \stackrel{\text{def}}{=} \mathcal{O}(10x)$ and $\mathcal{O}'''(x) \stackrel{\text{def}}{=} \mathcal{O}(11x)$.

  Below we use oracle $\mathcal{O}'''$ for the basic scheme $\mathcal{S}$, oracle $\mathcal{O}''$ for the CS-proofs, and oracle $\mathcal{O}'$ for our evasive relation. We note that if $\mathcal{O}$ is an ℓ_{out}-oracle, then so are $\mathcal{O}', \mathcal{O}''$ and $\mathcal{O}'''$.

- The universal function ensemble $\mathcal{U}$ from Subsection 4.2, with proper complexity bound $t(n) = n^{\log n}$. We denote by $M_{\mathcal{U}}$ the universal machine that decides the relation $R^{\mathcal{U}}$. That is, on input $(\langle i, s\rangle, y)$, machine $M_{\mathcal{U}}$ invokes the i^{th} evaluation algorithm, and accepts if $f_s^i(\langle i, s\rangle) = y$.

  We note that $M_{\mathcal{U}}$ works in time t in the worst case. More importantly, if $\mathcal{F}^i$ is a function ensemble that can be computed in time $p_i(\cdot)$ (where p_i is some polynomial), then for any strings s, y, on input $(\langle i, s\rangle, y)$, machine $M_{\mathcal{U}}$ works for only $\text{poly}(|i|) \cdot p_i(|s|)$ many steps.[14]

Using all the above, we describe an ideal signature scheme $\mathcal{S}'_u = (G, S'_u, V'_u)$. As usual, the key generation algorithm, G, remains unchanged. The signature and verification algorithms proceed as follows.

$S'^{\mathcal{O}}_u(\text{sk}, \text{msg}) \stackrel{\text{def}}{=}$

1. Parse msg as $\langle i, s, \pi\rangle$, and set $x = \langle i, s\rangle$ and $y = \mathcal{O}'(x)$. Let $n = |(x, y)|$.
2. Apply $\text{Ver}^{\mathcal{O}''}$ to verify whether π is a valid CS-proof, with respect to the oracle $\mathcal{O}''$ and security parameter 1^{n+k}, for the claim that the machine $M_{\mathcal{U}}$ accepts the input (x, y) within time $t(n)$.

   (The punch-line is that we do not directly check whether the machine $M_{\mathcal{U}}$ accepts the input (x, y) within time $t(n)$, but rather only if π is a valid CS-proof of this claim. Although $t(n) = n^{\log n}$, this CS-proof can be verified in polynomial-time.)
3. If π is a valid proof, then output (sk, msg).
4. Otherwise, output $S^{\mathcal{O}'''}(\text{sk}, \text{msg})$.

$V'^{\mathcal{O}}_u(\text{vk}, \text{msg}, \sigma) \stackrel{\text{def}}{=}$

1+2. As above

3. If π is a valid proof, then `accept`
4. Otherwise, output $V^{\mathcal{O}'''}(\text{vk}, \text{msg}, \sigma)$.

The computation required in Item 2 of the signature and verification algorithms can be executed in polynomial-time. The reason being that (by definition) verifying a CS-proof can be done in polynomial-time, provided the statement can be decided in at most exponential time (which is the case here since we have $t(n) = O(n^{\log n})$). It is also easy to see that for every pair (sk, vk) output by G, and for every msg and every $\mathcal{O}$, the string $S'^{\mathcal{O}}_u(\text{sk}, \text{msg})$ constitutes a valid signature of msg relative to vk and the oracle $\mathcal{O}$.

To show that the scheme is secure in the Random Oracle Model, we first observe that on security parameter 1^k it is infeasible to find a string x so that $(x, \mathcal{O}'(x)) \in R^{\mathcal{U}}$, since $R^{\mathcal{U}}$ is evasive. By Proposition 2.8, it is also infeasible to find (x, π) such that $(x, \mathcal{O}'(x)) \notin R_{\mathcal{U}}$ and yet π is a valid CS-proof of the contrary relative to $\mathcal{O}''$ (with security parameter $1^{|x|+\ell_{\text{out}}(k)+k}$). Thus, it is infeasible for a polynomial-time adversary to find a message that would pass the test on Item 2 of

[14] The point is merely that, for every fixed i, the expression $\text{poly}(|i|) \cdot p_i(|s|)$ is bounded by a polynomial in $|s|$.

the signature/verification algorithms above, and so we infer that the modified signature is secure in the Random Oracle Model.

We now show that for every candidate implementation, $\mathcal{F}$, there exists a polynomial-time adversary effecting total break via a chosen message attack (or, analogously, an existential forgery via a "key only" attack). First, for each function $f_s \in \mathcal{F}$, denote $f'_s(x) \stackrel{\text{def}}{=} f_s(01x)$, $f''_s(x) \stackrel{\text{def}}{=} f_s(10x)$, and $f'''_s(x) \stackrel{\text{def}}{=} f_s(11x)$. Then denote by $\mathcal{F}'$ the ensemble of the f'_s functions.

Suppose that $\mathcal{F}'$ is the i^{th} function ensemble in the enumeration mentioned above, namely $\mathcal{F}' = \mathcal{F}^i$. Given a randomly chosen k-bit seed s, the adversary generates a message $\text{msg} = \langle i, s, \pi \rangle$ so that π is a CS-proof (w.r.t the adequate security parameter) for the *true* statement that $M_{\mathcal{U}}$ accepts the input (x, y) within $t(|x| + |y|)$ steps, where $x = \langle i, s \rangle$ and $y = f'_s(x)$. Recall that the above statement is indeed true (since $f'_s \equiv f^i_s$), and hence the adversary can generate a proof for it in time which is polynomial in the time that it takes to compute f^i_s. (By the perfect completeness property of the CS-proof system, the ability to prove correct statements holds for *any* choice of the random oracle, and in particular when it is equal to f''_s.) Since this adversary is specifically designed to break the scheme in which the random oracle is implemented by $\mathcal{F}$, then the index i – which depends only on the choice of $\mathcal{F}$ – can be incorporated into the program of this adversary.

By the efficiency condition of CS-proofs, it is possible to find π (given an oracle access to f''_s) in time polynomial in the time that it takes $M_{\mathcal{U}}$ to accept the input (x, y). Since $\mathcal{F}^i$ is polynomial-time computable, then $M_{\mathcal{U}}$ works on the input $(x, y) = (\langle i, s \rangle, y)$ in polynomial time, and thus the described adversary also operates in polynomial-time.

By construction of the modified verification algorithm, ϵ is a valid signature on $\text{msg} = \langle i, s, \pi \rangle$, and so existential forgery is feasible a-priori. Furthermore, requesting the signer to sign the message msg yields the signing key, and thus total forgery. ■

Remark 4.5 It is immaterial for the above argument whether CS-proofs can be implemented in the "real world" (i.e., without access to random oracles). Specifically, it doesn't matter if one can cheat when the oracle is substituted by a candidate function ensemble, as in this case (i.e., in the real world implementation) it is sufficient for the adversary to invoke the proof system on valid statements. We do rely, however, on the perfect completeness of CS-proofs that implies that valid statements can be proven for any possible choice of oracle used in the proof system.

4.4 Encryption

The construction presented for signature schemes can be adapted to public-key encryption schemes in a straightforward way, yielding the following theorem:[15]

Theorem 4.6

(a) *Assume that there exists a public key encryption scheme that is semantically secure in the Random Oracle Model. Then there exists a public key encryption scheme that is semantically secure in the Random Oracle Model but is not semantically secure when implemented with any function ensemble.*[16]

(b) *Assume that there exists a public key encryption scheme that is secure under adaptive chosen ciphertext attack in the Random Oracle Model. Then there exists a scheme that is secure*

[15] Similarly, we can adapt the argument to shared-key (aka private-key) encryption schemes. See Remark 4.8.

[16] Here we refer to semantic security as defined by Goldwasser and Micali in [21], and not to the seemingly weaker definition presented in [16, 17]. Goldwasser and Micali allow the message space to depend on the public-key, whereas this is not allowed in [16, 17].

under adaptive chosen ciphertext attack in the Random Oracle Model, but implementing it with any function ensemble yields a scheme that is not semantically secure, and in which a chosen ciphertext attack reveals the secret decryption key.

Proof: In this proof we use the same notations as in the proof of Theorem 4.4. Let $\mathcal{E} = (G, E, D)$ be an encryption scheme that is semantically secure in the Random Oracle Model, and we modify it to get another scheme $\mathcal{E}' = (G, E', D')$. The key generation algorithm remains unchanged, and the encryption and decryption algorithms utilize a random oracle $\mathcal{O}$, which is again viewed as three oracles $\mathcal{O}', \mathcal{O}''$ and $\mathcal{O}'''$.

Modified encryption, $E'_{\text{ek}}{}^{\mathcal{O}}(\text{msg})$, of plaintext msg using the public encryption-key ek:

1. Parse msg as $\langle i, s, \pi \rangle$, set $x = \langle i, s \rangle$ and $y = \mathcal{O}'(x)$, and let $n = |(x, y)|$.
2. If π is a valid CS-proof, w.r.t oracle $\mathcal{O}''$ and security parameter 1^{n+k}, for the assertion that $M_{\mathcal{U}}$ accepts the pair (x, y) within $t(n)$ steps, then output $(1, \text{msg})$.
3. Otherwise (i.e., π is not such a proof), output $(2, E_{\text{ek}}^{\mathcal{O}'''}(\text{msg}))$.

Modified decryption, $D'_{\text{dk}}{}^{\mathcal{O}}(c)$, of ciphertext c using the private decryption-key dk:

1. If $c = (1, c')$, output c' and halt.
2. If $c = (2, c')$, output $D_{\text{dk}}^{\mathcal{O}'''}(c')$ and halt.
3. If $c = (3, c')$ then parse c' as $\langle i, s, \pi \rangle$, and set $x = \langle i, s \rangle$, $y = \mathcal{O}'(x)$, and $n = |(x, y)|$. If π is a valid CS-proof, w.r.t oracle $\mathcal{O}''$ and security parameter 1^{n+k}, for the assertion that $M_{\mathcal{U}}$ accepts the pair (x, y) within $t(n)$ steps, then output dk and halt.
4. Otherwise output ϵ.

The efficiency of this scheme follows as before. It is also easy to see that for every pair (ek, dk) output by G, and for every plaintext msg, the equality $D'_{\text{dk}}{}^{\mathcal{O}}(E'_{\text{ek}}{}^{\mathcal{O}}(\text{msg})) = \text{msg}$ holds for every $\mathcal{O}$. To show that the scheme is secure in the Random Oracle Model, we observe again that it is infeasible to find a plaintext that satisfies the condition in Item 2 of the encryption algorithm (resp., a ciphertext that satisfies the condition in Item 3 of the decryption algorithm). Thus, the modified ideal encryption scheme (in the Random Oracle Model) inherits all security features of the original scheme.

Similarly, to show that replacing the random oracle by any function ensemble yields an insecure scheme, we again observe that for any such ensemble there exists an adversary who – given the seed s – can generate a plaintext msg that satisfies the condition in Item 2 of the encryption algorithm. Hence, such an adversary can identify when msg is being encrypted (thus violates semantic security). This proves Part (a) of the theorem. For Part (b), the adversary generates a ciphertext c that meets the condition in Item 3 of the decryption algorithm, and ask for a decryption of c, thus obtaining the secret decryption key. ■

Remark 4.7 As opposed to Theorem 4.4, here we need to make computational assumptions, namely, that there exist schemes that are secure in the Random Oracle Model. (The results in [25] imply that it is unlikely that such schemes are proven to exist without making any assumptions.) Clearly, any scheme that is secure without random oracles is also secure in the Random Oracle Model. Recall that the former exist, provided trapdoor permutations exist [21, 36].

Remark 4.8 The constructions presented above can be adapted to yield many analogous results. For example, a result analogous to Theorem 4.6 holds for shared-key (aka private-key) encryption schemes. In this case no computational assumptions are needed since secure shared-key encryption is known to exist in the Random Oracle Model. Similarly, we can prove the existence of a CS-proof in the Random Oracle Model that has no implementations (via any function ensemble). In fact, as remarked in the Introduction, the same technique can be applied to practically any cryptographic application.

5 Restricted ensembles and other directions

Faced with the negative result of Theorem 3.4, one may explore restricted (and yet possibly useful) versions of "an implementation of a random oracle". One possibility is to put more stringent constraints on the use of the ensemble in a cryptographic scheme, and then to show that as long as the ensemble is only used in this restricted manner, it is guaranteed to maintain some aspects of correlation intractability.

In particular, notice that the proof of Theorem 3.4 relies heavily on the fact that the input to f_s can be as long as the seed s. Thus, one option would be to require that f_s be used only on inputs that are shorter than s. Specifically, require that each function f_s will only be applied to inputs of length $\ell_{\text{in}}(|s|)$, where $\ell_{\text{in}} : \mathsf{N} \to \mathsf{N}$ is some pre-specified function (e.g. $\ell_{\text{in}}(k) = k/2$). This leads to the corresponding restricted notion of correlation intractability (which is derived from Definition 3.2):

Definition 5.1 (restricted correlation intractability) *Let $\ell_{\text{in}}, \ell_{\text{out}} : \mathsf{N} \to \mathsf{N}$ be length functions. A machine M is called ℓ_{in}-respecting if $|M(s)| = \ell_{\text{in}}(|s|)$ for all $s \in \{0,1\}^*$.*

- *A binary relation R is evasive with respect to $(\ell_{\text{in}}, \ell_{\text{out}})$ if for any ℓ_{in}-respecting probabilistic polynomial-time oracle machine M*

$$\Pr_{\mathcal{O}}[x \leftarrow M^{\mathcal{O}}(1^k),\ (x, \mathcal{O}(x)) \in R] \quad = \quad \text{negl}(k)$$

  where $\mathcal{O} : \{0,1\}^{\ell_{\text{in}}(k)} \to \{0,1\}^{\ell_{\text{out}}(k)}$ is a uniformly chosen function and $\text{negl}(\cdot)$ *is a negligible function.*

- *We say that an ℓ_{out}-ensemble $\mathcal{F}$ is $(\ell_{\text{in}}, \ell_{\text{out}})$-restricted correlation intractable (or just ℓ_{in}-*correlation intractable*, for short), if for every ℓ_{in}-respecting probabilistic polynomial-time machine M and every evasive relation R w.r.t. $(\ell_{\text{in}}, \ell_{\text{out}})$, it holds that*

$$\Pr_{s \in \{0,1\}^k}[x \leftarrow M(s),\ (x, f_s(x)) \in R] = \text{negl}(k)$$

Weak ℓ_{in}-correlation intractability *is defined analogously by considering only polynomial-time recognizable R's.*

Most of this section is dedicated to demonstrating impossibility results for restricted correlation intractable ensembles, in some cases. We also highlight cases where existence of restricted correlation intractable ensembles is left as an open problem.

5.1 On the non-existence of restricted correlation intractable ensembles

The proof ideas of Theorem 3.4 can be easily applied to rule out the existence of certain restricted correlation intractable ensembles where the seed is too short.

Proposition 5.2

(a) *If $\ell_{\text{in}}(k) \geq k - O(\log k)$ for infinitely many k's, then there exists no ensemble that is $(\ell_{\text{in}}, \ell_{\text{out}})$-restricted correlation intractable, even in the weak sense.*

(b) *If $\ell_{\text{in}}(k)+\ell_{\text{out}}(k) \geq k+\omega(\log k)$, there exists no ensemble that is $(\ell_{\text{in}}, \ell_{\text{out}})$-restricted correlation intractable.*

Proof: The proof of (a) is a straightforward generalization of the proof of Theorem 3.4. Actually, we need to consider two cases: the case $\ell_{\text{in}}(k) \geq k$ and the case $k - O(\log k) \leq \ell_{\text{in}}(k) < k$. In the first case, we proceed as in the proof of Theorem 3.4 (except that we define $R^{\mathcal{F}} \stackrel{\text{def}}{=} \{(x, f_s(x)) : s \in \{0,1\}^*, x = s0^{\ell_{\text{in}}(|s|)-|s|}\}$). In the second case, for every ensemble $\mathcal{F}$, we define the relation

$$R^{\mathcal{F}} \stackrel{\text{def}}{=} \{(x, f_{xz}(x)) : x, z \in \{0,1\}^*, \ |x| = \ell_{\text{in}}(|xz|)\}$$

We show that $R^{\mathcal{F}}$ is evasive by showing that, for every $k \in \mathsf{N}$ and $x \in \{0,1\}^{\ell_{\text{in}}(k)}$, there exist at most polynomially (in k) many y's such that $(x, y) \in R^{\mathcal{F}}$. This is the case since $(x, y) \in R^{\mathcal{F}}$ implies that there exists some z such that $\ell_{\text{in}}(|xz|) = |x|$ and $y = f_{xz}(x)$. But using the case hypothesis we have $|x| = \ell_{\text{in}}(|xz|) \geq |xz| - O(\log |xz|)$, implying that $|z| = O(\log(|xz|))$ and hence also $|z| = O(\log |x|)$. Next, using the other case hypothesis (i.e., $k > \ell_{\text{in}}(k) = |x|$), we conclude that $|z| = O(\log k)$. Therefore, there could be at most polynomially many such z's, and so the upper bound on the number of y's paired with x follows. The evasiveness of $R^{\mathcal{F}}$ as well as the assertion that $R^{\mathcal{F}}$ is polynomial-time computable follow (assuming that the function ℓ_{in} itself is polynomial-time computable). On the other hand, consider the machine M that, on input s, outputs the $\ell_{\text{in}}(|s|)$-bit prefix of s. Then, for every $s \in \{0,1\}^*$, we have $(M(s), f_s(M(s))) \in R^{\mathcal{F}}$.

For the proof of (b), assume that $\ell_{\text{in}}(k) < k$ (for all but finitely many k's). We start by defining the "inverse" of the ℓ_{in} function

$$\ell_{\text{in}}^{-1}(n) \stackrel{\text{def}}{=} \min\{k \ : \ \ell_{\text{in}}(k) = n\}$$

(where, in case there exists no k such that $\ell_{\text{in}}(k) = n$, we define $\ell_{\text{in}}^{-1}(n) = 0$). By definition it follows that $k \geq \ell_{\text{in}}^{-1}(\ell_{\text{in}}(k))$, for all k's (because k belongs to the set $\{k' \ : \ \ell_{\text{in}}(k') = \ell_{\text{in}}(k)\}$), and that $\ell_{\text{in}}(\ell_{\text{in}}^{-1}(n)) = n$, whenever there exists some k for which $n = \ell_{\text{in}}(k)$. Next we define

$$R^{\mathcal{F}} \stackrel{\text{def}}{=} \left\{(x, f_{xz}(x)) \ : \ x, z \in \{0,1\}^*, \ |x| + |z| = \ell_{\text{in}}^{-1}(|x|)\right\}$$

This relation is well defined since, by the conditions on the lengths of x and z, we have $\ell_{\text{in}}(|xz|) = \ell_{\text{in}}(\ell_{\text{in}}^{-1}(|x|)) = |x|$ and so the function f_{xz} is indeed defined on the input x. In case $\ell_{\text{in}}(k) \leq k - \omega(\log k)$, this relation may not be polynomial-time recognizable. Still, it is evasive w.r.t. $(\ell_{\text{in}}, \ell_{\text{out}})$, since with security parameter k we have for every $x \in \{0,1\}^{\ell_{\text{in}}(k)}$

$$\begin{aligned} \left|\left\{y \in \{0,1\}^{\ell_{\text{out}}(k)} \ : (x,y) \in R^{\mathcal{F}}\right\}\right| &= \left|\left\{f_{xz}(x) \ : |z| = \ell_{\text{in}}^{-1}(\ell_{\text{in}}(k)) - \ell_{\text{in}}(k)\right\} \ \cap \ \{0,1\}^{\ell_{\text{out}}(k)}\right| \\ &\leq 2^{\ell_{\text{in}}^{-1}(\ell_{\text{in}}(k)) - \ell_{\text{in}}(k)} \\ &\leq 2^{k-\ell_{\text{in}}(k)} \end{aligned}$$

Using $k - \ell_{\text{in}}(k) \leq \ell_{\text{out}}(k) - \omega(\log k)$, we conclude that the set of y's paired with x forms a negligible fraction of $\{0,1\}^{\ell_{\text{out}}(k)}$, and so that $R^{\mathcal{F}}$ is evasive. Again, the machine M, that on input s outputs the $\ell_{\text{in}}(|s|)$-bit prefix of s, satisfies $(M(s), f_s(M(s))) \in R^{\mathcal{F}}$, for all s's. ■

Open problems. Proposition 5.2 still leaves open the question of existence of $(\ell_{\rm in}, \ell_{\rm out})$-restricted correlation intractable ensembles, for the case $\ell_{\rm in}(k) + \ell_{\rm out}(k) < k + O(\log k)$.[17] We believe that it is interesting to resolve the situation either way: Either provide negative results also for the above special case, or provide a plausible construction. Also open is the sub-case where $\ell_{\rm in}(k) + \ell_{\rm out}(k) = k + \omega(\log k)$ but one considers only *weak* $(\ell_{\rm in}, \ell_{\rm out})$-restricted correlation intractability. (Recall that Case (b) of Proposition 5.2 is proven using relations that are not known to be polynomial-time recognizable.)

5.2 Other limitations of restricted correlation intractable ensembles

Proposition 5.2 does not rule out the existence of correlation intractable ensembles having sufficiently long seed. This section demonstrates that even if such ensembles exist, then they are very non-robust constructs. Specifically, even if the ensemble $\mathcal{F} = \{f_s : |s| = k\}_k$ is restricted correlation intractable with respect to some length functions $(\ell_{\rm in}, \ell_{\rm out})$, the ensemble that is obtained by applying many independent copies of $\mathcal{F}$ and concatenating the results may not be. That is, for $m : \mathsf{N} \to \mathsf{N}$, define

$$\mathcal{F}^m \stackrel{\rm def}{=} \{f'_{\langle s_1,\dots,s_{m(k)}\rangle} : |s_1| = \cdots = |s_{m(k)}| = k\}_{k \in \mathsf{N}}\,, \tag{3}$$

where, for $\langle x_1, \dots, x_{m(k)}\rangle \in \{0,1\}^{m(k)\cdot\ell_{\rm in}(k)}$,

$$f'_{\langle s_1,\dots,s_{m(k)}\rangle}(\langle x_1, \dots, x_{m(k)}\rangle) \stackrel{\rm def}{=} \langle f_{s_1}(x_1), \dots., f_{s_{m(k)}}(x_{m(k)})\rangle\,. \tag{4}$$

Then, for sufficiently large m (e.g., $m(k) \geq k/\ell_{\rm in}(k)$ will do), the "direct product" ensemble $\mathcal{F}^m$ is not correlation intractable (not even in the restricted sense). That is,

Proposition 5.3 *Let $\ell_{\rm in}, \ell_{\rm out} : \mathsf{N} \to \mathsf{N}$ be length functions so that $\ell_{\rm in}(k) \leq k$, and let $m : \mathsf{N} \to \mathsf{N}$ be a polynomially-bounded function so that $m(k) \geq k/\ell_{\rm in}(k)$. Let $\mathcal{F}$ be an arbitrary function ensemble, and $\mathcal{F}^m$ be as defined in Eq. (3) and (4). Then, $\mathcal{F}^m$ is not correlation intractable, not even in the $(\ell^m_{\rm in}, \ell^m_{\rm out})$-restricted sense, where $\ell^m_{\rm in}(m(k)\cdot k) \stackrel{\rm def}{=} m(k)\cdot\ell_{\rm in}(k)$ and $\ell^m_{\rm out}(m(k)\cdot k) \stackrel{\rm def}{=} m(k)\cdot\ell_{\rm out}(k)$.*

Proof: We assume, for simplicity that $m(k) = k/\ell_{\rm in}(k)$ (and so $\ell_{\rm in}(k) = k/m(k)$ and $\ell^m_{\rm in}(m(k)\cdot k) = k$). Given $\mathcal{F}^m$ as stated, we again adapt the proof of Theorem 3.4. This time, using $\ell_{\rm in}(k) \leq k$, we define the relation

$$R^{\mathcal{F}^m} \stackrel{\rm def}{=} \bigcup_k \{\, (s, \langle f_s(s'), u\rangle) \;:\; |s| = k,\;\; s' \text{ is the } \ell_{\rm in}(k)\text{-prefix of } s,\;\; |u| = (m(k)-1)\cdot\ell_{\rm out}(k) \,\}$$

Notice that in this definition we have $|s| = \frac{k}{\ell_{\rm in}(k)}\cdot\ell_{\rm in}(k) = m(k)\cdot\ell_{\rm in}(k) = \ell^m_{\rm in}(m(k)\cdot k)$, and also $|f_s(s')| + |u| = m(k)\cdot\ell_{\rm out}(k) = \ell^m_{\rm out}(m(k)\cdot k)$, so this relation is indeed $(\ell^m_{\rm in}, \ell^m_{\rm out})$-restricted.

Again, it is easy to see that $R^{\mathcal{F}}$ is polynomial-time recognizable, and it is evasive since every string $x \in \{0,1\}^k$ is coupled with at most a $2^{-\ell_{\rm out}(k)}$ fraction of the possible $(m(k)\cdot\ell_{\rm out}(k))$-bit long strings, and $\ell_{\rm out}(k) = \omega(\log k) = \omega(\log(m(k)\cdot k))$. (Here we use the hypothesis $m(k) = \mathrm{poly}(k)$.)

On the other hand, consider a (real-life) adversary that given the seed $\overline{s} = \langle s_1, \dots, s_{m(k)}\rangle \in \{0,1\}^{m(k)\cdot k}$ for the function $f'_{\langle s_1,\dots,s_{m(k)}\rangle}$, sets the input to this function to be equal to s_1. Denoting the $\ell_{\rm in}(k)$-prefix of s_1 (equiv., of $\overline{s}$) by s'_1, it follows that $f_{s_1}(s'_1)$ is a prefix of $f'_{\langle s_1,\dots,s_{m(k)}\rangle}(s_1)$ and so $(s_1, f'_{\langle s_1,\dots.,s_{m(k)}\rangle}(s_1)) \in R^{\mathcal{F}}$. Thus, this real-life adversary violates the (restricted) correlation intractability of $\mathcal{F}^m$. ■

[17] In fact such ensembles do exist in case $k \geq 2^{\ell_{\rm in}(k)}\cdot\ell_{\rm out}(k)$ (since the seed may be used to directly specify all the function's values), but we dismiss this trivial and useless case.

5.3 On correlation intractability for multiple invocations

Proposition 5.2 relates only to forming rare relationships between a *single* input-output pair. This section demonstrates that, if one generalizes the definition of correlation intractability to consider also evasive relations over long sequences of inputs and outputs, then the negative result in Proposition 5.2 can be extended for arbitrary ℓ_{in} and ℓ_{out}. That is:

Definition 5.4 (multiple-input restricted correlation intractability) *Let $\ell_{\mathrm{in}}, \ell_{\mathrm{out}} : \mathsf{N} \to \mathsf{N}$ be length functions. We consider probabilistic polynomial-time oracle machines that on input 1^k have oracle access to a function $\mathcal{O} : \{0,1\}^{\ell_{\mathrm{in}}(k)} \to \{0,1\}^{\ell_{\mathrm{out}}(k)}$.*

- *A relation R over pairs of binary sequences is* evasive with respect to $(\ell_{\mathrm{in}}, \ell_{\mathrm{out}})$ (or $(\ell_{\mathrm{in}}, \ell_{\mathrm{out}})$-evasive) *if for any probabilistic polynomial-time machine M as above, there exists a negligible function* negl *such that*

$$\Pr_{\mathcal{O}}\left[(x_1,...,x_m) \leftarrow M^{\mathcal{O}}(1^k)\,;\quad \begin{array}{l} |x_1| = \ldots = |x_m| = \ell_{\mathrm{in}}(k) \\ \text{and } ((x_1,...,x_m),(\mathcal{O}(x_1),...,\mathcal{O}(x_m)) \in R \end{array}\right] = \mathrm{negl}(k)$$

  As usual, $\mathcal{O} : \{0,1\}^{\ell_{\mathrm{in}}(k)} \to \{0,1\}^{\ell_{\mathrm{out}}(k)}$ is a uniformly chosen function.

- *We say that an ℓ_{out}-ensemble $\mathcal{F}$ is $(\ell_{\mathrm{in}}, \ell_{\mathrm{out}})$-*restricted multiple-input correlation intractable *if for every $(\ell_{\mathrm{in}}, \ell_{\mathrm{out}})$-evasive relation R and every probabilistic polynomial-time machine M, there exists a negligible function* negl *such that*

$$\Pr_{s \in \{0,1\}^k}\left[(x_1,...,x_m) \leftarrow M(s)\,;\quad \begin{array}{l} |x_1| = \ldots = |x_m| = \ell_{\mathrm{in}}(k) \\ \text{and } ((x_1,...,x_m),(f_s(x_1),...,f_s(x_m)) \in R \end{array}\right] = \mathrm{negl}(k)$$

Proposition 5.5 *Let $\ell_{\mathrm{in}}, \ell_{\mathrm{out}} : \mathsf{N} \to \mathsf{N}$ be arbitrary length functions, with $\ell_{\mathrm{in}}(k) \geq 2 + \log k$ and $\ell_{\mathrm{out}}(k) \geq 1$. Then there exist no $(\ell_{\mathrm{in}}, \ell_{\mathrm{out}})$-restricted* multiple-input *correlation intractable function ensembles.*

Proof: For simplicity, we consider first the case $\ell_{\mathrm{out}}(k) \geq 2$. Let $\mathcal{F}$ be an ℓ_{out}-ensemble. Adapting the proof of Theorem 3.4, we define the relation

$$R^{\mathcal{F}} \stackrel{\mathrm{def}}{=} \bigcup_k \left\{ ((x_1,\ldots,x_k),(f_s(x_1),\ldots,f_s(x_k))) : \begin{array}{l} x_i = (i, s_i), \text{ with } s_i \in \{0,1\} \\ \text{and } s = s_1 \ldots s_k \end{array} \right\} \tag{5}$$

(Notice that since $\ell_{\mathrm{in}}(k) > 1 + \log k$, the x_i's are indeed in the range of the function f_s.) Clearly, this relation is polynomial-time recognizable. To see that this relation is evasive, notice that for any fixed k-bit seed $s = s_1 \ldots s_k$, we have

$$\Pr_{\mathcal{O}}[\mathcal{O}(i,s_i) = f_s(i,s_i) \text{ for } i = 1 \ldots k] = 2^{-\ell_{\mathrm{out}}(k)\cdot k}$$

Hence, the probability that there exists a seed s for which $\mathcal{O}(i,s_i) = f_s(i,s_i)$ holds, for $i = 1,...,k$, is at most $2^k \cdot 2^{-\ell_{\mathrm{out}}(k)\cdot k} \leq 2^{-k}$. It follows that

$$\Pr_{\mathcal{O}}[\exists x_1,...,x_k \;\; ((x_1,\ldots,x_k),(\mathcal{O}(x_1),\ldots,\mathcal{O}(x_k))) \in R^{\mathcal{F}}] \leq 2^{-k}$$

However, the corresponding multiple-input restricted correlation intractability condition does not hold: For any $s = s_1 \ldots s_k \in \{0,1\}^k$, setting $x_i = (i, s_i)$ we get $((x_1,...,x_k),(f_s(x_1),...,f_s(x_k))) \in R^{\mathcal{F}}$.

To rule out the case $\ell_{\mathrm{out}}(k) = 1$, we redefine $R^{\mathcal{F}}$ so that $((x_1,...,x_{2k}),(f_s(x_1),...,f_s(x_{2k}))) \in R^{\mathcal{F}}$ if $x_i = (i, s_i)$ for $i = 1,...,k$ and $x_i = (i, 0)$ for $i = k+1,...,2k$. ■

5.4 Implications for signatures and encryption

The results from Section 5.3 can be used to extend the negative results from Theorems 4.4 and 4.6 also to the case of restricted ensembles with short seeds. These theorems and proofs are very similar to the ones from Section 4. Here we only state the theorem for signatures and provide a proof sketch.

Theorem 5.6 *There exists a signature scheme that is existentially unforgeable under a chosen message attack in the Random Oracle Model, but such that when implemented with any restricted function ensemble with $\ell_{\text{in}}(k) \geq \log k + \omega(1)$, the resulting scheme is existentially forgeable using key-only attack and totally breakable under chosen message attack.*

Proof Sketch: Fix some length functions $(\ell_{\text{in}}, \ell_{\text{out}})$, with $\ell_{\text{in}}(k) \geq \log k + \omega(1)$ (and assume, for simplicity, that $\ell_{\text{out}}(k) \geq 2$). We follow the same three steps as in the proof of Theorem 4.4, with the following modifications: For the first step, the message to be signed is parsed as a vector $x = \langle x_1 \dots x_n \rangle$ (with $|x_i| = \ell_{\text{in}}(k)$ when the security parameter is k) and the signer checks whether the sequence $(\langle x_1, \dots, x_n \rangle, \langle \mathcal{O}(x_1), \dots, \mathcal{O}(x_k) \rangle)$ stands in the relation $R^{\mathcal{F}}$ from Eq. (5). For the second step we again use enumeration of ensembles (this time, however, we enumerate $(\ell_{\text{in}}, \ell_{\text{out}})$-restricted ensembles). The "universal ensemble" that we need can be defined as

$$R^{\mathcal{U}} \stackrel{\text{def}}{=} \bigcup_{k,m} \left\{ \Big(\langle x_1 \dots x_{k+m} \rangle, \langle f_s^i(x_1) \dots f_s^i(x_{k+m}) \rangle \Big) : \begin{array}{l} x_j = (j, y_j), \text{ where } y_j \in \{0,1\}, \text{ and} \\ \text{for } j = 1, ..., m, \ y_j \text{ is the } j\text{th bit of } \langle i \rangle \\ \text{for } j = 1, ..., k, \ y_{m+j} \text{ is the } j\text{th bit of } s \end{array} \right\}$$

where $\langle i \rangle$ is the encoding of the i'th ensemble, $\mathcal{F}^i$. Note that since $\ell_{\text{in}}(k) \geq \log k + \omega(1)$, then for each fixed i, the input length will conform to the length restriction eventually (i.e., for a large enough security parameter). The third step uses CS-proofs, just as in the proof of Theorem 4.4. □

5.5 On weak restricted correlation-intractable ensembles

In all our negative results, the evasive relation demonstrating that a certain function ensemble is not correlation-intractable is more complex than the function ensemble itself. A natural restriction on correlation-intractability is to require that it holds only for relations recognizable within certain fixed polynomial time bounds (or some fixed space bound), and allowing the function ensemble to have a more complex polynomial-time evaluation algorithm. We stress that, in both the definition of evasiveness and correlation-intractability, the adversary that generates the inputs to the relation is allowed arbitrary (polynomial) running time; this time may be larger than both the time to evaluate the function ensemble and the time to evaluate the relation. Such a restricted notion of correlation-intractability may suffice for some applications, and it would be interesting to determine whether function ensembles satisfying it do exist. Partial results in this direction were obtained by Nissim [31] and are described next:

Proposition 5.7 ([31]) *Let $\ell_{\text{in}}, \ell_{\text{out}} : \mathsf{N} \to \mathsf{N}$ be arbitrary length functions, with $k \geq \ell_{\text{out}}(k) \cdot (\ell_{\text{in}}(k) + \omega(\log k))$.*[18] *Then, for every binary relation R that is evasive with respect to $(\ell_{\text{in}}, \ell_{\text{out}})$ and recognizable in polynomial-time, there exists a function ensemble $\mathcal{F}^R = \{f_s\}$ that is correlation-intractable with respect to R; that is, for every ℓ_{in}-respecting probabilistic polynomial-time machine*

[18] Recall that $(\ell_{\text{in}}, \ell_{\text{out}})$-restricted correlation-intractable ensembles exist for $k \geq 2^{\ell_{\text{in}}(k)} \cdot \ell_{\text{out}}(k)$; see Footnote 17.

M it holds that

$$\Pr_{s\in\{0,1\}^k}[x \leftarrow M(s),\;\; (x, f_s(x)) \in R] = \mathrm{negl}(k)$$

We note that the postulated construction uses a seed length that is longer than $\ell_{\mathrm{in}} + \ell_{\mathrm{out}}$. Thus, this positive result capitalizes on both restrictions discussed above (i.e., both the length and the complexity restrictions).

Proof: Let $t = \ell_{\mathrm{in}}(k) + \omega(\log k)$. For every seed $s = (s_1, ..., s_t) \in \{0,1\}^{t\cdot\ell_{\mathrm{out}}(k)}$, we define $f_s : \{0,1\}^{\ell_{\mathrm{in}}(k)} \to \{0,1\}^{\ell_{\mathrm{out}}(k)}$ so that $f_{s_1,...,s_t}(x)$ equals s_i if i is the smallest integer such that $(x, s_i) \notin R$. In case $(x, s_i) \in R$ holds for all i's, we define $f_{s_1,...,s_t}(x)$ arbitrarily.

Let $R(x) \stackrel{\mathrm{def}}{=} \{y : (x,y) \in R\}$, and $S_k \stackrel{\mathrm{def}}{=} \{x \in \{0,1\}^{\ell_{\mathrm{in}}(k)} : |R(x)| \leq 2^{\ell_{\mathrm{out}}(k)}/2\}$ (S stands for "Small image"). Since R is evasive, it is infeasible to find an $x \in \{0,1\}^{\ell_{\mathrm{in}}(k)}$ not in S_k. Thus, for every probabilistic polynomial-time M, $\Pr_{s\in\{0,1\}^k}[M(s) \notin S_k] = \mathrm{negl}(k)$. On the other hand, the probability that such $M(s)$ outputs an $x \in S_k$ so that $(x, f_s(x)) \in R$ is bounded above by[19]

$$\begin{aligned}
\Pr_{s\in\{0,1\}^k}[\exists x \in S_k \text{ s.t. } (x, f_s(x)) \in R] &\leq \Pr_{s\in\{0,1\}^k}[\exists x \in S_k\, \forall i\;\; (x, s_i) \in R] \\
&\leq |S_k| \cdot \max_{x\in S_k}\left\{\Pr_s[\forall i\;\; (x, s_i) \in R]\right\} \\
&\leq 2^{\ell_{\mathrm{in}}(k)} \cdot (1/2)^t \;=\; \mathrm{negl}(k)
\end{aligned}$$

Combining the two cases, the proposition follows. ■

Considering the notion of multiple-input correlation-intractability when restricting the complexity of the relation (and allowing the function ensemble to be more complex), Nissim has obtained another impossibility result [31]:

Proposition 5.8 ([31]) *There exists an evasive relation R that is recognizable in polynomial-time so that no function ensemble $\mathcal{F} = \{f_s\}$ is multiple-input correlation-intractable with respect to R; that is, for every function ensemble $\mathcal{F} = \{f_s\}$ there exists a polynomial-time machine M such that*

$$\Pr_s[(x_1, ..., x_t) \leftarrow M(s)\,;\;\; ((x_1, ..., x_t), (f_s(x_1), ..., f_s(x_t))\in R\,] \;=\; 1$$

Furthermore, for some universal polynomial p, which is independent of $\mathcal{F}$, it holds that $t < p(|x_1|)$.

We stress that the above assertion includes even function ensembles that have (polynomial-time) evaluation algorithms of running time greater than the time it takes to recognize t-tuples of corresponding length in the relation. Furthermore, it includes function ensembles having seeds of length exceeding the total length of pairs in the relation.

Proof Sketch: We follow the ideas underlying the proof of Theorem 4.4. Specifically, using the universal machine $M_{\mathcal{U}}$ and the algorithms (PRV and VER) of a CS-proof system, we consider a relation R that contains pairs of binary sequences, so that $((x, \pi, q_1..., q_m), (y, \phi, a_1..., a_m)) \in R$ if these strings describe an accepting execution of the CS-verifier with respect to machine $M_{\mathcal{U}}$. That is, we require that the following conditions hold:

[19] For the first inequality, we use the fact that if there exists an i such that $(x, s_i) \notin R$ then $(x, f_s(x)) \notin R$.

1. All the strings $y, \phi, a_1 ..., a_m$ have the same length.[20] Below we denote this length by ℓ_{out}, $|y| = |\phi| = |a_1| = \cdots = |a_m| = \ell_{\text{out}}$.
2. The string π is an alleged CS-proof for the assertion that the machine $M_{\mathcal{U}}$ accepts the input (x, y) within $t(n) = n^{\log n}$ steps, where $n \stackrel{\text{def}}{=} |x| + |y|$.
3. Given access to an oracle that on queries q_i returns answers a_i, and given security parameter $n + \ell_{\text{out}}$ and input $w = (\langle M_{\mathcal{U}} \rangle, (x, y), t(n))$, the CS verifier VER accepts the CS-proof π after querying the oracle on $q_1 \dots q_m$ (in this order), and obtaining the corresponding answers $a_1 \dots a_m$.

   (Here we use the fact that the verifier is deterministic, and thus its queries are determined by its input and the answers to previous queries.)

Recall that, by definition, m is bounded by a fixed polynomial in n. In fact, in Micali's construction [27], m is poly-logarithmic in n. We comment that, assuming the existence of suitable collision-intractable hash functions, one may obtain $m = 1$ (cf. [28]. In addition, one may need to make some minor modification in the above construction.)

As in the proof of Theorem 4.4, using the computational soundness of CS-proofs, it can be shown that the above relation is evasive. By the additional efficiency conditions of CS-proofs, it follows that the relation is recognizable in polynomial-time. On the other hand, as in the proof of Theorem 4.4, for every function ensemble $\mathcal{F}^i = \{f_s^i\}$ there exists a polynomial-time adversary A, that on input s produces a sequence $(x, \pi, q_1, ..., q_m)$ so that $((x, \pi, q_1, ..., q_m), (f_s^i(x), f_s^i(\pi), f_s^i(q_1), ..., f_s^i(q_m))) \in R$. This is done as follows: First A sets $x = \langle i, s \rangle$, $y = f_s^i(x)$, and $n \stackrel{\text{def}}{=} |x| + |y|$. Next, A constructs a CS-proof that indeed $M_{\mathcal{U}}$ accepts (x, y) within $n^{\log n}$ steps, and sets π to equal this proof. (This step takes time polynomial in the evaluation time of $f_s^i(x)$.) Note that since (x, y) is indeed accepted by $M_{\mathcal{U}}$ (in less than $n^{\log n}$ steps), the verifier accept π as a proof no matter how the oracle is determined (since perfect completeness holds). Finally, the adversary invokes the verifier (on input consisting mainly of (x, y) and π), and (by emulating the oracle) determines interactively the oracle queries and answers of the verifier; that is, for every $j = 1, ..., m$, the adversary determines the j^{th} query made by the verifier, sets q_j to equal this query, and provides the verifier with the answer $f_s^i(q_j)$. □

5.6 On the failure of some specific constructions

We conclude our study of restricted correlation-intractable by pointing out the failure of several natural candidates, even in the restricted case of single-invocation.

Pseudorandom function ensembles. A natural conjecture is that pseudorandom functions, as defined in [19], may yield at least some restricted form of correlation-intractability. However, it is easy to see that this conjecture fails miserably. For example, for any (super-logarithmic) $\ell_{\text{in}}, \ell_{\text{out}} : \mathsf{N} \to \mathsf{N}$ (possibly, $\ell_{\text{in}}(k) + \ell_{\text{out}}(k) \ll k$), consider a pseudorandom function ensemble $\{f_s : \{0,1\}^{\ell_{\text{in}}(|s|)} \to \{0,1\}^{\ell_{\text{out}}(|s|)}\}_{s \in \{0,1\}^*}$. Then, defining $f'_s(x) = 0^{\ell_{\text{out}}(|s|)}$ if x is a prefix of s, and $f'_s(x) = f_s(x)$ otherwise, yields a pseudorandom function ensemble that is clearly not correlation-intractability (even in the restricted case of single-invocation).[21]

[20] The string ϕ is a "don't care" value that serves as a place holder for the output of the function on input π.

[21] We have assumed, for simplicity, that $\ell_{\text{in}}(k) \le k$, which is the more interesting case anyhow. Otherwise, the exceptional input for f'_s is set to be $s0^{\ell_{\text{in}}(|s|) - |s|}$.

We mention that even the pseudorandom function ensembles that results from the construction of Goldreich, Goldwasser and Micali [19] are not necessarily correlation-intractable. Specifically, one can construct a pseudorandom generator such that applying to it the construction from [19] results in a pseudorandom function ensemble, in which given the seed one can efficiently find an input that is mapped to the all-zero string [18]. We stress that this holds for any $\ell_{\text{in}}(k) = \ell_{\text{out}}(k) = \omega(\log k)$.

Universal and higher independence hash functions. Lastly, we point out that, in general, collections of t-wise independent hashing functions are not correlation-intractable (even in a restricted sense). For example, consider the collection of t-wise independent hash functions derived from the set of degree $t-1$ polynomials over $\mathrm{GF}(2^n)$. Specifically, for each such (degree $t-1$) polynomial $p : \mathrm{GF}(2^n) \to \mathrm{GF}(2^n)$, consider the function $f : \{0,1\}^n \to \{0,1\}^{n/2}$ that results by letting $f(x)$ be the $n/2$-bit prefix of $p(x)$. Note that this collection has seed length $t \cdot n$, which is much larger than the sum of the lengths of the input and output (i.e., $n + (n/2)$). Still, given the description of such a function (i.e., the polynomial p) it is easy to find an input that is mapped to $0^{n/2}$ (e.g., by selecting uniformly $r \in \{0,1\}^{n/2}$ and finding a root of the polynomial $p(x) - \alpha$, where $\alpha = 0^{n/2} r$).

6 Conclusions

The results in this work show conclusively that the random oracle methodology is not sound, in general, with respect to the natural notions of "implementation of the random oracle." Although these negative results seem to have no effect on the security of common practical schemes that were built using this methodology, it should serve as a warning sign.

At the same time, one can view these results as a challenge: Is it possible to find a "reasonable notion of implementation", relative to which one can show the soundness of this methodology (at least, in some interesting cases)? The work of Canetti [8] is a first step in that direction, but more steps seem to be called for. In particular, one could consider a more general notion of "implementation", as a compiler that takes a scheme that works in the Random Oracle Model and produces a scheme that works in the standard model (i.e., without a random oracle). This compiler may do more than just replace the oracle queries by function calls. However, such a general-purpose compiler is ruled out by Nielsen's recent work [30], which shows that there are (natural) cryptographic tasks that can be securely realized in the Random Oracle Model but cannot be securely realized in the standard model. Thus, one should focus on special-purpose compilers (i.e., compilers that can be applied only to restricted classes of schemes and/or tasks). Furthermore, the compiler should preserve the complexity of the original Random Oracle Model scheme (as done by the straightforward compiler that just replaces the oracle queries by function calls).[22]

Regarding the implications of our results to the current practice of the Random Oracle Methodology, the authors have different opinions. Rather than trying to strike a mild compromise, we prefer to present our disagreements in the most controversial form.

6.1 Ran's Conclusions

Real-life cryptographic applications are complex objects. On top of the "cryptographic core," these applications typically involve numerous networking protocols, several other applications, user-interfaces, and in fact also an entire operating-system. The security of an application depends on

[22]In fact, removing the complexity-preservation requirement may allow trivial compilers that ignore the given Random Oracle Model scheme and just return a hard-wired scheme for the standard model.

the security of all these components operating in unison. Thus, in principle, the best way to gain assurance in the security of a cryptographic application is to analyze it as a single unit, bones and feathers included.

However, analyzing an entire system is prohibitively complex. Moreover, we often feel that the "essence" of a cryptographic application can be presented in a relatively simple way without getting into many details that, we feel, are "extraneous" to the actual security. Consequently, we often make *abstractions* of a cryptographic application by leaving many details "outside the model". Such abstractions are indeed essential tools in protocol analysis. Nonetheless, great caution is needed when making abstractions: While sound abstractions are important and useful, unsound abstractions can be dangerous and misleading. Thus, it is crucial to make sure that one uses a sound abstraction, namely one that helps us distinguish between good and bad applications.

One popular abstraction is to treat computers in a network as interactive Turing machines who run one specific (and relatively simple) algorithm, and assume that delivery of messages is done simply by having one machine write values on the tapes of another machine. We are then satisfied with defining and analyzing security of a protocol in this abstract model. In other words, this abstraction implicitly uses the following methodology (which I'll call the "Interactive Turing machine methodology"): Design and analyze a protocol in the "idealized system" (i.e., using Turing machines). Next, come up with an "implementation" of the idealized protocol by adding the components that deal with the networking protocols, the operating system, the user interfaces, etc. Now, "hope" that the implementation is indeed secure.

We widely believe that this methodology is sound, in the sense that if an idealized protocol is secure then there *exist* secure implementations of it. Furthermore, security of an idealized protocol is a good predictor for the feasibility of finding a good implementation to it. (Of course, finding secure implementations to secure idealized protocols is a far-from-trivial task, and there is probably no single automatic method for securely implementing any idealized protocol. But this does not undermine the soundness of the "Interactive Turing machine methodology".)

The Random Oracle methodology is, in essence, another proposed abstraction of cryptographic applications. It too proposes to define and analyze security of protocols in an idealized model, then perform some transformation that is "outside the formal model", and now "hope" that the resulting implementation is secure. At first it looks like a great abstraction: It does away with specific implementation issues of "cryptographic hash functions" and concentrates on designing protocols assuming that an "ideal hash function" is available. Indeed, many protocols that were designed using this methodology are remarkably simple and efficient, while resisting all known attacks.

However, as shown in this work, and in sharp contrast to the "Interactive Turing machine methodology", the Random Oracle Methodology is not sound. Furthermore, it is a bad predictor to the security of implementations: Not only do there *exist* idealized protocols that have no secure implementations, the methods described in this work can be used to turn practically *any* idealized protocol described in the literature into a protocol that remains just as secure in the idealized model, but has no secure implementations. This leaves us no choice but concluding that, in spite of its apparent successes, the Random Oracle model is a bad abstraction of protocols for the purpose of analyzing security.

The loss of reductions to hard problems. The above discussion should provide sufficient motivation to be wary of security analyses in the Random Oracle Model. Nonetheless, let me highlight the following additional disturbing aspect of such analysis.

One of the great contributions of complexity-based modern cryptography, developed in the past quarter of a century, is the ability to base the security of many varied protocols on a small number of well-defined and well-studied complexity assumptions. Furthermore, typically the proof of security of a protocol provides us with a method for transforming adversary that breaks the security of the said protocol into an adversary that refutes one of the well-studied assumptions. In light of our inability to prove security of protocols from scratch, this methodology provides us with the "next best" evidence for the security of protocols.

The Random Oracle Methodology does away with these advantages. Assume that an idealized protocol A is proven secure in the Random Oracle Model based on, say, the Diffie-Hellman assumption, and that someone comes up with a way to break *any implementation* of A. This does not necessarily mean that it is now possible to break Diffie-Hellman! Consequently, the reducibility of the security of A to the hardness of Diffie-Hellman is void. This brings us back to a situation where the security of each protocol is a "stand-alone" conjecture and is, in essence, unrelated to the hardness of known problems.

Possible alternative directions. In spite of its shortcomings, the Random Oracle Methodology seems to generate simple and efficient protocols against which no attacks are known. Consequently, the Random oracle model can be regarded as a good initial idealized setting for designing and analyzing protocols. Still, it must be kept in mind that analysis in the random oracle model is only a first step towards meaningful security analysis. It does not by itself provide any security guarantees for the implementations in the standard model.

One possible direction towards providing formal justification for some of the protocols constructed using the Random Oracle methodology, is to identify useful, special-purpose properties of the random oracle, which can be also provided by a fully specified function (or function ensemble) and so yield secure implementations of certain useful ideal systems. First steps in this direction were taken in [8, 11, 15]. Hopefully, future works will push this direction further.

6.2 Oded's Conclusions

My starting point is that within the domain of science, every deduction requires a rigorous justification.[23] In contrast, unjustified deductions should not be allowed; especially not in a subtle research area such as Cryptography. Furthermore, one should refrain from making statements that are likely to mislead the listener/reader, such as claiming a result in a restricted model while creating the impression that it holds also in a less restricted model. The presentation of such a result should clearly state the restrictions under which it holds, and refrain from creating the impression that the result extends also to a case where these restrictions are waived (unless this is indeed true (and one can prove it)). Needless to say, it is perfectly OK to conjecture that a restricted result extends also to a case when these restrictions are waived, but the stature of such a statement (as a conjecture) should be clear.

The above abstract discussion directly applies to security in the Random Oracle Model. Deducing that the security of a scheme in the Random Oracle Model means anything about the security of its implementations, without proper justification, is clearly wrong. This should have been clear also before the current work. It should have also been clear that no proper justification of a deduction from security in the Random Oracle Model to security of implementations has ever been given. The contributions of the current work are two-fold:

[23] This does not disallow creative steps committed in the course of research, without proper justification. Such unjustified steps are the fuel of progress. What I refer to are claims that are supposed to reflect valid facts. Such claims should be fully justified, or offered as conjectures.

1. This work uncovers inherent difficulties in the project of providing conditions that would allow (justifiable) deduction from security in the Random Oracle Model to security of implementations. Such a project could have proceeded by identifying properties that characterize proofs of security in the Random Oracle Model, and (justifiably) deducing that the such schemes maintain their security when implemented with ensembles satisfying these properties. The problem with this project is that correlation intractability should have been (at the very least) one of these properties, but (as we show) no function ensemble can satisfy it.

2. As stated above, deducing that the security of a scheme in the Random Oracle Model means anything about the security of its implementations, without proper justification, is clearly wrong. The current work presents concrete examples in which this unjustified deduction leads to wrong conclusions. That is, it is shown that not only that unjustified deduction regarding the Random Oracle Model MAY *lead to wrong conclusions*, but rather than in some cases INDEED this unjustified deduction DOES *lead to wrong conclusions*. Put in other words, if one needs a concrete demonstration of the dangers of unjustified deduction when applied to the Random Oracle Model, then this work provides it.

The bottom-line: It should be clear that the Random Oracle Methodology is not sound; that is, the mere fact that a scheme is secure in the Random Oracle Model cannot be taken as evidence (or indication) to the security of (possible) implementations of this scheme. Does this mean that the Random Oracle Model is useless? Not necessarily: it may be useful as a test-bed (or as a sanity check).[24] Indeed, if the scheme does not perform well on the test-bed (resp., fails the sanity check) then it should be dumped. But one should *not* draw wrong conclusions from the mere fact that a scheme performs well on the test-bed (resp., passes the sanity check). In summary, the Random Oracle Methodology is actually a method for ruling out *some* insecure designs, but this method is not "complete" (i.e., it may fail to rule out insecure designs).[25]

6.3 Shai's Conclusions

The negative results in this work (and in particular Theorems 4.4 and 4.6) leave me with an uneasy feeling: adopting the view that a good theory should be able to explain "the real world", I would have liked theoretical results that explain the apparent success of the random oracle methodology in devising useful, seemingly secure, cryptographic schemes. (Indeed, this was one of the original motivations for this work.) Instead, in this work we show that security of cryptographic schemes in the Random Oracle Model does not necessarily imply security in "the real world". Trying to resolve this apparent mismatch, one may come up with several different explanations. Some of those are discussed below:

- The current success of this methodology is due to pure luck: all the current schemes that are proven secure in the Random Oracle Model, happen to be secure also in the "real world" for

[24] This explains the fact the Random Oracle Methodology is in fact used in practice. In also explains why many reasonable schemes, the security of which is still an open problem, are secure in the Random Oracle Model: good suggestions should be expected to pass a sanity check.

[25] Would I, personally, endorse this method is a different question. My answer is very much time-sensitive: Given the current misconceptions regarding the Random Oracle Model, I would suggest not to include, in currently published work, proofs of security in the Random Oracle Model. My rationale is that the dangers of misconceptions (regarding such proofs) seem to out-weight the gain of demonstrating that the scheme passed a sanity check. I hope that in the future such misconceptions will be less prevailing, at which time it would be indeed recommended to report on the result of a sanity check.

no reason. However, our "common sense" and sense of aesthetics must lead us to reject such explanation.

- The current apparent success is a mirage: some of the schemes that are proven secure in the Random Oracle Model are not really secure, and attacks on them may be discovered in the future.

  This explanation seems a little more attractive than the previous one. After all, a security proof in the Random Oracle Model eliminates a broad class of potential attacks (i.e., the ones that would work also in the Random Oracle Model), and in many cases it seems that attacks of this type are usually the ones that are easier to find. Hence, it makes sense that if there exists a "real life" attack on a scheme that is secure in the Random Oracle Model, it may be harder – and take longer – to find this attack.

- Another possible explanation is that the random oracle methodology works for the current published schemes, due to some specific features of these schemes that we are yet to identify. That is, maybe it is possible to identify interesting classes of schemes, for which security in the Random Oracle Model implies the existence of a secure implementation.[26]

  Identifying such interesting classes, and proving the above implication, is an important – and seemingly hard – research direction. (In fact, it even seems to be hard to identify classes of schemes for which this implication makes a reasonable conjecture.)

As we illustrate in the introduction, we could attribute the "mismatch" between the apparent security of the practical schemes that were devised using the random oracle methodology and the (proven) insecurity of our contrived schemes, to our current lack of knowledge regarding "what can be done with a code of a function". One can hope that improving the understanding of this point could shed light also on the relations between the security of ideal schemes and their implementations. (I.e., let us either find new types of attacks, as per the second point above, or identify cases where attacks are infeasible, as per the third point.)

For now, however, I view the random oracle methodology as a very useful "engineering tool" for devising schemes. As a practical matter, I would much rather see today's standards built around schemes that are proven secure in the Random Oracle Model, than around schemes for which no such proofs exist. If nothing else, it makes finding attacks on such schemes a whole lot harder.

Acknowledgments

We wish to thank Silvio Micali for enlightening discussions. We thank Clemens Holenstein for uncovering a flaw in an earlier version of the proof of Proposition 5.5. Special thanks to Kobbi Nissim for telling us about his related results (i.e., Propositions 5.7 and 5.8) and permitting us to include them in this write-up. We also thank the reviewers for their useful comments.

References

[1] B. Barak. How to go beyond the black-box simulation barrier. In *42rd Annual Symposium on Foundations of Computer Science*, pages 106–115. IEEE, 2001.

[26] One particularly silly example are schemes that do not use the oracle. Another, more interesting example, are schemes that only use the "perfect one-way" property of the oracle; see [8, 11].

[2] B. Barak. "constant-round coin-tossing with a man in the middle or realizing the shared random string model". In *43th Annual Symposium on Foundations of Computer Science*. IEEE, 2002. to appear.

[3] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of software obfuscation. In *Advances in Cryptology – CRYPTO '01*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18. Springer-Verlag, 2001.

[4] M. Bellare, R. Canetti and H. Krawczyk. Keying Hash Functions for Message Authentication. In *CRYPTO'96*, Springer LNCS (Vol. 1109), pages 1–15.

[5] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *1st Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.

[6] M. Bellare and P. Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In *Advances in Cryptology - EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer-Verlag, 1996.

[7] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13:850–864, 1984.

[8] R. Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *Advances in Cryptology - CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 455–469. Springer-Verlag, 1997.

[9] R. Canetti, U. Feige, O. Goldreich and M. Naor. Adaptively Secure Computation. *28th Symposium on Theory of Computing (STOC),* ACM, 1996. Fuller version in MIT-LCS-TR #682, 1996.

[10] R. Canetti, O. Goldreich and S. Halevi. The Random Oracle Methodology, Revisited. Preliminary version in *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing*, Dallas, TX, May 1998. ACM. TR version(s) available on-line from `http://eprint.iacr.org/1998/011` and `http://xxx.lanl.gov/abs/cs.CR/0010019`.

[11] R. Canetti, D. Micciancio and O. Reingold. Perfectly one-way probabilistic hashing. In *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing*, pages 131–140, Dallas, TX, May 1998. ACM.

[12] I. Damgård. Collision free hash functions and public key signature schemes. In *Advances in Cryptology - EUROCRYPT'87*, volume 304 of *Lecture Notes in Computer Science*, pages 203–216. Springer-Verlag, 1987.

[13] C. Dwork, M. Naor, O. Reingold, and L. Stockmeyer. Magic functions. In *40th Annual Symposium on Foundations of Computer Science*, pages 523–534. IEEE, 1999.

[14] A. Fiat and A. Shamir. How to prove yourself. practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–189. Springer-Verlag, 1986.

[15] R. Gennaro, S. Halevi and T. Rabin. Secure Hash-and-Sign Signatures Without the Random Oracle. In *Advances in Cryptology - EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 123–139. Springer-Verlag, 1999.

[16] O. Goldreich. A Uniform Complexity Treatment of Encryption and Zero-Knowledge. *Journal of Cryptology*, Vol. 6, No. 1, pages 21–53, 1993.

[17] O. Goldreich. *Encryption Schemes – fragments of a chapter.* December 1999. Available from `http://www.wisdom.weizmann.ac.il/~oded/foc-vol2.html`

[18] O. Goldreich. The GGM Construction does NOT yield Correlation Intractable Function Ensembles. *ECCC* (`http://www.eccc.uni-trier.de/eccc/`), TR02-047, July 2002.

[19] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):210–217, 1986.

[20] O. Goldreich, and H. Krawczyk. On the Composition of Zero-Knowledge Proof Systems. *SIAM Journal on Computing*, 25(1):169–192, 1996.

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

[22] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, 17(2):281–308, Apr. 1988.

[23] L. Guillou and J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessors minimizing both transmission and memory. In *Advances in Cryptology - EUROCRYPT'88*, volume 330 of *Lecture Notes in Computer Science*, pages 123–128. Springer-Verlag, 1988.

[24] S. Hada and T. Tanaka. A relationship between one-wayness and correlation intractability. In *Proceedings of the 2nd International Workshop on Practice and Theory in Public Key Cryptography (PKC'99)*, volume 1560 of *Lecture Notes in Computer Science*, pages 82–96, Japan, March 1999. Springer-Verlag.

[25] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 44–61, Seattle, WA, May 1989. ACM.

[26] J. Kilian. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the 24th Annual ACM Symposium on the Theory of Computing*, pages 723–732. ACM, May 1992.

[27] S. Micali. Computationally Sound Proofs. *SIAM Journal on Computing*, Vol. 30 (4), pages 1253–1298, 2000. Preliminary version in *35th FOCS*, 1994.

[28] M. Naor and K. Nissim. Computationally sound proofs: Reducing the number of random oracle calls. Manuscript, 1999.

[29] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 33–43, 1989.

[30] J.B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *Advances in Cryptology – CRYPTO'02*, to appear.

[31] K. Nissim. Two results regarding correlation intractability. Manuscript, 1999.

[32] T. Okamoto. Provably secure and practical identification scheme and corresponding signature scheme. In *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53. Springer-Verlag, 1992.

[33] D. Pointcheval and J. Stern. Security proofs for signature schemes. In *Advances in Cryptology - EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer-Verlag, 1996.

[34] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 387–394, 1990.

[35] C. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[36] A. C. Yao. Theory and applications of trapdoor functions. In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91. IEEE, Nov. 1982.