Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology*

Ueli Maurer, Renato Renner, and Clemens Holenstein

Department of Computer Science, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland
{maurer,renner,holenste}@inf.ethz.ch

Abstract. The goals of this paper are two-fold. First we introduce and motivate a generalization of the fundamental concept of the indistinguishability of two systems, called indifferentiability. This immediately leads to a generalization of the related notion of reducibility of one system to another. In contrast to the conventional notion of indistinguishability, indifferentiability is applicable in settings where a possible adversary is assumed to have access to additional information about the internal state of the involved systems, for instance the public parameter selecting a member from a family of hash functions.

Second, we state an easily verifiable criterion for a system $\mathcal{U}$ not to be reducible (according to our generalized definition) to another system $\mathcal{V}$ and, as an application, prove that a random oracle is not reducible to a weaker primitive, called asynchronous beacon, and also that an asynchronous beacon is not reducible to a finite-length random string. Each of these irreducibility results alone implies the main theorem of Canetti, Goldreich, and Halevi stating that there exist cryptosystems that are secure in the random oracle model but for which replacing the random oracle by any implementation leads to an insecure cryptosystem.

1 Introduction

1.1 Motivation: Cryptographic Security Proofs

The following generic methodology is often applied in cryptographic security proofs. To prove the security of a cryptosystem $\mathcal{C}(\cdot)$ with access[1] to a (real) component system $\mathcal{S}$, denoted $\mathcal{C}(\mathcal{S})$, one first proves that the system $\mathcal{C}(\mathcal{T})$ is secure for some idealized component system $\mathcal{T}$. Second, one proves the following general relation between $\mathcal{S}$ and $\mathcal{T}$: For *any* cryptosystem $\tilde{\mathcal{C}}(\cdot)$, the security of $\tilde{\mathcal{C}}(\mathcal{T})$ is not affected if $\mathcal{T}$ is replaced by $\mathcal{S}$. Let us consider two examples.

* This research was supported by SNF Project No. 20-66716.01.

[1] The notation $\mathcal{C}(\cdot)$ means that $\mathcal{C}$ takes as an argument (or is connected to) a system that replies to queries by $\mathcal{C}$.

Example 1. Let $\mathcal{T}$ be a source of truly random bits (secret for two communicating parties A and B) and let $\mathcal{S}$ be a pseudo-random bit generator (with secret key shared by A and B). If $\mathcal{C}(\cdot)$ denotes XOR-based encryption (i.e., $\mathcal{C}(\mathcal{T})$ denotes the one-time pad and $\mathcal{C}(\mathcal{S})$ denotes an additive stream cipher with key-stream generator $\mathcal{S}$), then the security of $\mathcal{C}(\mathcal{S})$ follows from the security of $\mathcal{C}(\mathcal{T})$ and the fact that, for any efficient distinguisher (or adversary), $\mathcal{S}$ behaves essentially like $\mathcal{T}$, i.e., $\mathcal{S}$ and $\mathcal{T}$ are (computationally) indistinguishable.

Example 2. Let $\mathcal{T}$ be a random oracle $\mathcal{R}$, (i.e., a publicly accessible random function) and let $\mathcal{S}$ be a hash function $\mathcal{H}(\mathcal{F})$, where $\mathcal{H}$ is a hash algorithm depending on a public parameter $\mathcal{F}$ (selecting one function from a class of functions). In contrast to pseudo-randomness (where the parameter is secret), no hash function can implement a random oracle in the above sense, as proved by Canetti, Goldreich, and Halevi [6]. In other words, there exists a cryptosystem $\mathcal{C}(\cdot)$ such that $\mathcal{C}(\mathcal{R})$ is secure while $\mathcal{C}(\mathcal{H}(\mathcal{F}))$ is insecure for any hash algorithm $\mathcal{H}$.

It is important to note that the formalization of this second example is more involved than the first. Obviously, a random oracle is easily distinguishable from a hash function if one knows its program and the public parameter, but this fact does not prove the above mentioned claim that a random oracle can generally not be replaced by a hash function. What then is needed to prove this claim and, more generally, similar impossibility results? It is the purpose of this paper to formalize this problem and to provide the answer.

1.2 Random Oracles, Beacons, and Other Systems

In this paper, we will be concerned with the following general question: For given systems $\mathcal{S}$ and $\mathcal{T}$, can $\mathcal{T}$ be replaced by $\mathcal{S}$ in the above sense? A natural extension of this question is whether a system $\mathcal{U}$ can be reduced to a system $\mathcal{V}$, i.e., whether there exists an efficient algorithm $\mathcal{B}$ such that $\mathcal{U}$ can be replaced by $\mathcal{B}(\mathcal{V})$ (in the above sense).

One example of such a system that we will consider more closely is the random oracle. Its importance in cryptography is due to the so called random oracle methodology where the security of a cryptosystem is proven under the assumption that a common randomly chosen function (the *random oracle*) is accessible by each party. This fact is then used as evidence for the security of the corresponding (real) cryptosystem where the random oracle is replaced by a hash function. The methodology was first made explicit by Bellare and Rogaway [2] and has been used in many papers (e.g. [8, 9, 17, 13, 2, 11, 3, 16]).

A (binary) random oracle $\mathcal{R}$ can be seen as an infinite sequence $R_1, R_2, \ldots$ of public random bits where any arbitrary bit R_x can be accessed in one computational step. One can also think of weaker primitives where the cost to access the randomness is higher. In particular, we introduce a primitive, called *(binary) asynchronous beacon*[2] $\mathcal{Q}$, defined as a sequence of random bits $R_1, R_2, \ldots$ which

[2] The term "beacon", due to Rabin, is used here only in the sense described. In particular, the fact that for Rabin's beacons the randomness is available simultaneously

can only be read sequentially, i.e., the time needed to access R_x is linear in x. A natural question is whether one can implement a random oracle using an asynchronous beacon, i.e., whether there is an efficient algorithm $\mathcal{B}$ such that $\mathcal{B}(\mathcal{Q})$ behaves like $\mathcal{R}$. (Note that for each input, $\mathcal{B}$ could make polynomially many queries to $\mathcal{Q}$ before generating the output.)

An even weaker primitive is a *finite random string* $\mathcal{F}$, i.e., a finite sequence of bits $R_1, \ldots, R_n$ (e.g., accessible in constant time). One could also consider other systems between a finite random string, an asynchronous beacon, and a random oracle, for which the random bits might be accessible faster than sequentially but not in an arbitrary (random access) manner, or where the distribution of the random bits is not uniform. In a sense, a random oracle and a finite random string are two extreme points on a scale, and an asynchronous beacon is somewhere in the middle.

For any two such systems $\mathcal{U}$ and $\mathcal{V}$ one can still ask the question whether $\mathcal{U}$ can be implemented using $\mathcal{V}$. This paper formalizes and solves this problem. We show that, loosely speaking, the answer to this question is characterized by the rates at which entropy can be accessed in the systems $\mathcal{U}$ and $\mathcal{V}$. As special cases one sees that a random oracle cannot be implemented using an asynchronous beacon, and a beacon cannot be implemented using a finite random string. This also proves the main result of [6] as a simple consequence of the fact that a random oracle $\mathcal{R}$ contains substantially more entropy than a finite random string $\mathcal{F}$, in a manner to be made precise.

1.3 Indistinguishability and Indifferentiability

Informally, two systems $\mathcal{S}$ and $\mathcal{T}$ are said to be indistinguishable if no (efficient) algorithm $\mathcal{D}(\cdot)$, connected to either $\mathcal{S}$ or $\mathcal{T}$, is able to decide whether it is interacting with $\mathcal{S}$ or $\mathcal{T}$. As mentioned above, the security of a cryptosystem $\mathcal{C}(\mathcal{S})$ involving a component $\mathcal{S}$ is typically proven by considering the cryptosystem $\mathcal{C}(\mathcal{T})$ obtained from $\mathcal{C}(\mathcal{S})$ where the component $\mathcal{S}$ is replaced by an idealized component $\mathcal{T}$. The original system $\mathcal{C}(\mathcal{S})$ is secure if (a) the system $\mathcal{C}(\mathcal{T})$ is secure, and (b) the component $\mathcal{S}$ is indistinguishable from $\mathcal{T}$ (cf. Example 1).

The notion of reducibility is directly based on indistinguishability. A system $\mathcal{U}$ is said to be reducible to $\mathcal{V}$ if the system $\mathcal{V}$ can be used to construct a new system $\mathcal{B}(\mathcal{V})$ which is indistinguishable from $\mathcal{U}$. Again, reducibility is useful for cryptographic security proofs: If $\mathcal{U}$ is reducible to $\mathcal{V}$, then, for any cryptosystem $\mathcal{C}(\mathcal{U})$ using $\mathcal{U}$ as a component, there is another cryptosystem based on $\mathcal{V}$, namely $\mathcal{C}(\mathcal{B}(\mathcal{V}))$, having the same functionality and, in particular, providing the same security as $\mathcal{C}(\mathcal{U})$.

However, these considerations are all subject to the assumption that the party using such a component has exclusive access to it, i.e., that all other parties, including a possible adversary, are unable to directly influence the component's

---

to all parties, and that future beacon outputs remain secret until released, is not of relevance here.

behavior or obtain any information about its randomness. As described in Example 2, this is not the case for many components. Indeed, while for each party the output of a random oracle $\mathcal{R}$ is indistinguishable from the output of a local random function $\mathcal{R}^{\mathrm{loc}}$, the security of a cryptosystem based on $\mathcal{R}^{\mathrm{loc}}$ (where, e.g., the randomness is used for a randomized encryption) might obviously be lost when replacing this component by $\mathcal{R}$.

In order to extend the definition of indistinguishability such as to include this type of systems, we will propose a new concept of indistinguishability, called *indifferentiability*. Together with its derived notion of reducibility, it will allow for exactly the same general statements about the security of cryptosystems as the conventional definitions. In particular, this means that, first, if a component $\mathcal{S}$ is indifferentiable from $\mathcal{T}$, then the security of any cryptosystem $\mathcal{C}(\mathcal{T})$ based on $\mathcal{T}$ is not affected when replacing $\mathcal{T}$ by $\mathcal{S}$. Second, differentiability of $\mathcal{S}$ from $\mathcal{T}$ implies the existence of a cryptosystem $\mathcal{C}(\cdot)$ for which this replacement of components is not possible, i.e., $\mathcal{C}(\mathcal{T})$ is secure but becomes insecure if $\mathcal{T}$ is substituted by $\mathcal{S}$. Thus, similar to conventional indistinguishability, indifferentiability is the weakest possible property allowing for security proofs of the generic type described above, but it applies to more general settings.

1.4 Organization of the Paper

In Section 2, we give a straightforward proof of the classical separation result in [6] that a random oracle cannot be realized by a (family of) hash functions. While this separation result also follows directly from our general results derived in the subsequent sections, we think that starting with a self-contained proof of this (well-known) example will help the reader to understand the motivation for the definitions and to follow the rest of the paper. Section 4 and Section 5 are concerned with the generalization of the concept of indistinguishability, called indifferentiability, and the corresponding generalization of reducibility, respectively. These notions are then applied in Section 6 to state and prove a general irreducibility criterion, which is used in Section 7 to derive separation results for finite random strings, beacons, and random oracles.

2 A Motivating Example: A Simple Proof of the Impossibility of Implementing a Random Oracle

The following proposition directly implies the separation result as formulated in [6]. Its original proof is quite involved as it is based on techniques like Micali's CS-proofs [11]. Very recently, the same authors [7] showed that their result extends to signature schemes for only short messages. Other similar impossibility results are proposed in [12] and [1].

Proposition 1. *There exists a signature scheme $\mathcal{C}(\cdot)$ (consisting of a key-generating, a signing, and a verification algorithm) with access to either a random oracle $\mathcal{R}$ or an implementation thereof such that the following holds (with respect to some security parameter k):*

– *$\mathcal{C}(\mathcal{R})$ is secure, i.e., the probability that an attacker against $\mathcal{C}(\mathcal{R})$ is successful is negligible in k.*[3]
– *There is an adversary breaking $\mathcal{C}(f)$ for any arbitrary efficiently computable function f. In particular, $\mathcal{C}(\mathcal{H}(\mathcal{F}))$ is insecure for any hash function $\mathcal{H}$ with public parameter $\mathcal{F}$.*
– *$\mathcal{C}(\cdot)$ is efficient (i.e., the running time of the algorithms is polynomially bounded in the size of their input and the security parameter k).*

Proof. The proof consists of two parts. First, we construct $\mathcal{C}(\cdot)$ based on a distinguishing algorithm $\mathcal{D}(\cdot)$ which has the property that the behavior of $\mathcal{D}(\mathcal{R})$ is different from $\mathcal{D}(f)$. Second, we give a construction for $\mathcal{D}(\cdot)$ and prove that it has all the desired properties.

Let us thus assume that $\mathcal{D}(\cdot)$ is an algorithm taking as input a bitstring m (together with a security parameter k) and generating a binary output such that the following holds:

(a) The probability (over the randomness of $\mathcal{R}$) that there exists an input causing $\mathcal{D}(\mathcal{R})$ to output 1 is negligible in k.
(b) For any efficiently computable function f, there exists an input m causing $\mathcal{D}(f)$ to output 1. Moreover, m is easily computable given an algorithm for efficiently computing f.
(c) $\mathcal{D}(\cdot)$ is efficient (i.e., its running time is polynomially bounded by the size of its input m and the security parameter k).

Let $\bar{\mathcal{C}}(\cdot)$ be an efficient signature scheme which is secure when accessing a random oracle. The signature scheme $\mathcal{C}(\cdot)$ is then constructed by modifying the signing algorithm of $\bar{\mathcal{C}}(\cdot)$ as follows: On input m, it first calls $\mathcal{D}(\cdot)$ for input m. If $\mathcal{D}(\cdot)$ outputs 0, m is signed as usual (i.e., by calling the signing algorithm of $\bar{\mathcal{C}}(\cdot)$). Otherwise, it behaves completely insecurely (e.g., by revealing a secret key).

It is easy to see that $\mathcal{C}(\cdot)$ satisfies the requirements of the proposition: The security of $\mathcal{C}(\mathcal{R})$ follows directly from property (a). Furthermore, property (b) implies that there is an input m (efficiently computable by an adversary) causing $\mathcal{C}(f)$ to behave completely insecurely. Finally, the efficiency of $\mathcal{C}(\cdot)$ follows from the efficiency of $\mathcal{D}(\cdot)$ (property (c)) and the efficiency of $\bar{\mathcal{C}}(\cdot)$.

It remains to be proven that an algorithm $\mathcal{D}(\cdot)$ with the desired properties (a) to (c) indeed exists. We give an explicit construction for $\mathcal{D}(\cdot)$ and then show that properties (a) to (c) are satisfied. For the following, assume without loss of generality that the random oracle $\mathcal{R}$ is binary, i.e., its outputs are single bits.

Construction of $\mathcal{D}$ $\mathcal{D}(\cdot)$ interprets its input m as a pair (π, t) consisting of an encoding of a program π for a universal Turing machine and a unary encoding of some integer t (i.e., $t \leq |m|$). Let $q = 2|\pi| + k$ (where $|\pi|$ is the length of the encoding of π). For inputs $x = 1, \ldots, q$, $\mathcal{D}(\cdot)$ simulates at most t steps of

[3] A function $f : k \mapsto f(k)$ is *negligible* in k if $f(k)$ decreases faster than the inverse of any polynomial in k.

the program π, resulting in outcomes $\pi(1), \ldots, \pi(q)$.[4] Similarly, $\mathcal{D}(\cdot)$ sends the queries $x = 1, \ldots, q$ to the component it is connected to ($\mathcal{R}$ or f), resulting in answers $a(1), \ldots, a(q)$. If $\pi(x) = a(x)$ for all $x = 1, \ldots, q$, $\mathcal{D}(\cdot)$ outputs 1, and 0 otherwise.

$\mathcal{D}$ satisfies property (a). For any fixed program π, let p_π be the probability (over the randomness of $\mathcal{R}$) that for an input m encoding π, $\mathcal{D}(\mathcal{R})$ outputs 1. By construction, this happens if and only if $\pi(x) = a(x)$ for all $x = 1, \ldots, q$. Since, for each x, the random output $a(x)$ (of the binary random oracle $\mathcal{R}$) is equal to the output $\pi(x)$ (of the fixed program π) with probability at most $1/2$, we have $p_\pi \leq 2^{-q} = 2^{-2|\pi|-k}$. Hence, the probability p_l of the event that there exists a program π of length l such that $\mathcal{D}(\mathcal{R})$ outputs 1 is bounded by

$$p_l \leq \sum_{\pi \in \{0,1\}^l} p_\pi \leq 2^l \cdot 2^{-2l-k} = 2^{-l-k} \ .$$

Finally, the probability p that there exists a program π of arbitrary length causing $\mathcal{D}(\mathcal{R})$ to output 1 is bounded by

$$p \leq \sum_{l=1}^{\infty} p_l \leq \sum_{l=1}^{\infty} 2^{-l} \cdot 2^{-k} \leq 2^{-k} \ .$$

$\mathcal{D}$ satisfies property (b). Let π be an arbitrary program that efficiently computes f, and let t be the maximum running time of π for all inputs $y \in \{1, \ldots, q\}$ where $q = 2|\pi| + k$. By construction, the values $\pi(x)$ computed by $\mathcal{D}(f)$ on input $m := (\pi, t)$ satisfy $\pi(x) = f(x)$. Consequently, the equalities $\pi(x) = a(x)$ tested by $\mathcal{D}(f)$ hold for all values $x = 1, \ldots, q$, causing $\mathcal{D}(f)$ to output 1. Note that the maximum running time t can be determined efficiently given the program π (since π is efficient). The input m is thus efficiently computable from π.

$\mathcal{D}$ satisfies property (c). The running time of $\mathcal{D}(\mathcal{R})$ is essentially given by the time needed to compute the $q = 2|\pi| + k$ values $\pi(1), \ldots, \pi(q)$. For the computation of each of these values, the program π is executed for at most t steps. Since $|\pi|$ as well as the number t are both bounded by the size of m (recall that t is unary encoded in m), the running time of $\mathcal{D}(\mathcal{R})$ satisfies $O((2|\pi| + k) \cdot t) \leq O((|m| + k)^2)$. □

3 Basic Definitions and Notation

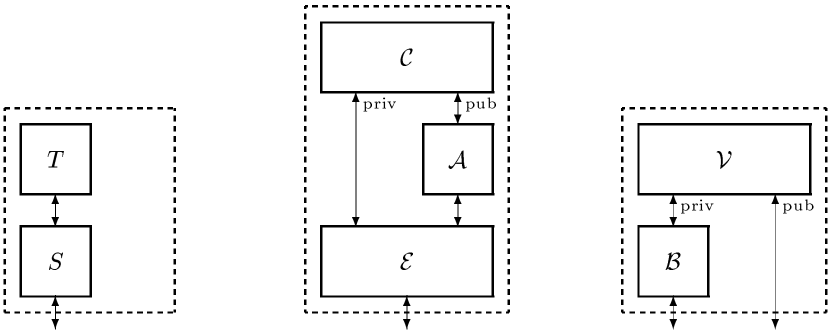
3.1 Interacting Systems

For the representation of (cryptographic) systems, we will basically adapt the terminology introduced in [10]. A $(\mathcal{X}, \mathcal{Y})$-*system* is a sequence of conditional

[4] If the program π does not generate an output after t steps, $\pi(i)$ is set to some dummy value.

probability distributions $P_{Y_i|X^iY^{i-1}}$ ($i \in \mathbb{N}$) with $X^i := [X_1, \dots, X_i]$ and $Y^{i-1} := [Y_1, \dots, Y_{i-1}]$, where X_i, called the *ith input*, and Y_i, the *ith output*, are random variables with range $\mathcal{X}$ and $\mathcal{Y}$, respectively. Intuitively speaking, a system is defined by the probability distribution of each output Y_i conditioned on all previous inputs X^i and outputs Y^{i-1}. If each output Y_i of $\mathcal{S}$ only depends on the actual input X_i, and possibly some randomness, then $\mathcal{S}$ is called a *random function*. For instance, a system $\mathcal{S}$ might be specified by an algorithm, where, for each input, the output is computed according to a given sequence of instructions. For convenience, we will assume that the systems' inputs and outputs are natural numbers, or, equivalently, their representation as finite bitstrings.

A *configuration of systems* is a set of systems where the systems' interfaces are pairwise connected. Any configuration of systems can be seen as a new system. For instance, let $\mathcal{S}$ be a system with two interfaces and let $\mathcal{T}$ be a system whose interface is connected to the first interface of $\mathcal{S}$. The resulting system, denoted as $\mathcal{S}(\mathcal{T})$, has one interface corresponding to the second (free) interface of $\mathcal{S}$ as shown in Fig. 1(a). In this case, the original system $\mathcal{S}$ is denoted as $\mathcal{S}(\cdot)$, and $\mathcal{T}$ is called *component* of $\mathcal{S}(\mathcal{T})$. More complex constructions are denoted similarly, e.g., $\mathcal{E}(\mathcal{C}^{\text{priv}}, \mathcal{A}(\mathcal{C}^{\text{pub}}))$ and $\mathcal{B}(\mathcal{V}^{\text{priv}})$ for the configuration depicted in Fig. 1(b) and Fig. 1(c), respectively.



(a) $\mathcal{S}(\mathcal{T})$ (b) $\mathcal{E}(\mathcal{C}^{\text{priv}}, \mathcal{A}(\mathcal{C}^{\text{pub}}))$ (c) $\mathcal{B}(\mathcal{V}^{\text{priv}})$

Fig. 1. Composition of systems.

Many complexity-theoretic and cryptographic properties of systems and particularly of algorithms are defined in terms of their asymptotic behavior with respect to some *security parameter* k. Thus, in the sequel, when speaking of a "system" $\mathcal{S}$, we will rather mean a family $(\mathcal{S}_k)_{k\in\mathbb{N}}$ parameterized by k, where each $\mathcal{S}_k$ is a system in the sense described above.

3.2 A Notion of Efficiency for Systems

An algorithm $\mathcal{B}$ is said to be *computationally efficient* if its running time is bounded by a polynomial in its input size and the security parameter k. Similarly to the computational efficiency of algorithms, we are interested in a certain notion of efficiency for systems $\mathcal{S}$ and constructions based on them. However, since a system $\mathcal{S}$ is not necessarily described by an algorithm, the usual formulation in terms of the number of computational steps is not sufficiently general. A more abstract approach to overcome this problem is to assign to each $(\mathcal{X},\mathcal{Y})$-system $\mathcal{S}$ a *cost function* c with range $\mathbb{R}^+$ specifying the amount of a certain resource (e.g. time), needed to process an input. For simplicity, we will assume that these costs only depend on the actual input, i.e., c is a function mapping elements from $\mathcal{X}$ to $\mathbb{R}^+$. Additionally, the costs c of a composite system $\mathcal{B}(\mathcal{V})$ must be compatible with the costs $\bar{c}$ of the underlying component $\mathcal{V}$, i.e., for any input x to $\mathcal{B}(\mathcal{V})$, $c(x)$ is at least as large as the sum of the costs $\bar{c}(\bar{x}_i)$ for all queries $\bar{x}_1,\ldots,\bar{x}_n$ sent by $\mathcal{B}$ to $\mathcal{V}$ while processing x.

Similarly to the usual notion of computational efficiency of algorithms, we say that a system $\mathcal{S}$ (or, more precisely, the class $(\mathcal{S}_k)_{k\in\mathbb{N}}$ of systems $\mathcal{S}_k$ with cost functions c_k) is *cost-efficient* if $c_k(x)$ is bounded by a polynomial in the input length $|x|$ and the security parameter k, i.e., $c_k(x) \leq p(|x|,k)$ for some polynomial p. For two systems $\mathcal{U}$ and $\mathcal{V}$, let $\Gamma(\mathcal{V}/\mathcal{U})$ be the set of all deterministic systems[5] $\mathcal{B}(\cdot)$ such that the costs of the system $\mathcal{B}(\mathcal{V})$ are bounded by a polynomial in the costs of the system $\mathcal{U}$ and the security parameter k. This means that, for any $\mathcal{B}(\cdot) \in \Gamma(\mathcal{V}/\mathcal{U})$, the construction $\mathcal{B}(\mathcal{V})$ is as cost-efficient (up to a polynomial factor) as $\mathcal{U}$, and, in particular, if the system $\mathcal{U}$ is cost-efficient, then so is the system $\mathcal{B}(\mathcal{V})$.

We will see in Section 6 that the entropy of the output of a system expressed in terms of the costs to produce this output is a measure allowing for deciding whether a certain reduction is possible. Let the system $\mathcal{S}_k$ be a random function with cost function c_k which is monotonically increasing in its inputs, and let $Y_1,\ldots,Y_{n_t}$ be the sequence of outputs of $\mathcal{S}_k$ on inputs $1,\ldots,n_t$, where n_t is the maximal input x such that $c_k(x) \leq t$. The functions $h^0_{\mathcal{S}_k}$ and $h^\infty_{\mathcal{S}_k}$ are defined, based on two different entropy measures, as

$$h^0_{\mathcal{S}_k}(t) := H_0(Y_1,\ldots,Y_{n_t}) \qquad \text{and} \qquad h^\infty_{\mathcal{S}_k}(t) := H_\infty(Y_1,\ldots,Y_{n_t}),$$

respectively, where $H_0(Z) := \log_2 |\mathcal{Z}|$, and where H_∞ is the min-entropy (defined as $H_\infty(z) := -\log_2 \max_{z\in\mathcal{Z}} P_Z(z)$). That is, for any bound t on the costs c_k determining a maximum input n_t, the quantities $h^0_{\mathcal{S}_k}(t)$ and $h^\infty_{\mathcal{S}_k}(t)$ measure the entropy of the outputs of the system $\mathcal{S}_k$ for inputs $1,\ldots,n_t$ (where the probability is taken over the internal randomness of $\mathcal{S}_k$). Clearly, $h^0_{\mathcal{S}}$ and $h^\infty_{\mathcal{S}}$ are monotonically increasing functions, and $h^0_{\mathcal{S}}(t) \geq h^\infty_{\mathcal{S}}(t)$.

[5] The restriction to deterministic systems $\mathcal{B}(\cdot)$ does not restrict the generality of our results. It simply implies that any randomness to be used by $\mathcal{B}(\cdot)$ must be modeled explicitly (by a random system attached to $\mathcal{B}(\cdot)$).

3.3 Cryptosystems and Security

A cryptosystem as well as any cryptographic primitive can generally be modeled as a random system providing interfaces to certain players. Usually, these players are either honest parties or controlled by an adversary. In this paper, we will be concerned with settings where the cryptographic primitives can be accessed by the honest players and the adversary in some predefined way. As an example, consider a publicly accessible resource (e.g., a random oracle or a public random string), where the interfaces to all players are identical. In this case, a possible adversary can access exactly the same information as the honest parties. Another example is a private resource, (e.g., a source of private randomness), to which the adversary is assumed to have no (direct) access at all.

In general, one might want to model situations where the adversary has some partial access to a cryptographic primitive. We thus define a *resource* $\mathcal{S}$ to be a random system with two interfaces, called *private* and *public*, respectively. In the following, we will think of the private and the public interface as being accessible by the honest parties and the adversary, respectively. A resource $\mathcal{S}$ is called *public* if the private and the public interface are identical (i.e., the answers to identical queries are identical).

Let $\mathcal{U}$ and $\mathcal{V}$ be resources. Similarly to the set $\Gamma(\mathcal{V}/\mathcal{U})$, we denote by $\Gamma^{\mathrm{p}}(\mathcal{V}/\mathcal{U})$ the set of deterministic systems $\mathcal{B}(\cdot)$ such that the costs of the system $\mathcal{B}(\mathcal{V}) := \mathcal{B}(\mathcal{V}^{\mathrm{priv}})$ resulting from connecting $\mathcal{B}(\cdot)$ to the private interface of $\mathcal{V}$ (cf. Fig. 1(c)) are polynomially bounded by the costs of $\mathcal{U}$ and a security parameter k.

In the following, we think of a *cryptosystem* $\mathcal{C}$ as being a resource (with a private and a public interface, modeling the access of the honest parties and the adversary, respectively). The security of a cryptosystem $\mathcal{C}$ is characterized relative to an ideal cryptosystem $\mathcal{C}'$ which *by definition* is secure. Obviously, this requires the ability to *compare* the security of cryptosystems, i.e., it needs to be specified what it means for a cryptosystem $\mathcal{C}$ to be *at least as secure* as another cryptosystem $\mathcal{C}'$. The following definition is based on ideas proposed by Canetti [4, 5], and by Pfitzmann and Waidner [14, 15] (for the case of static adversaries), adapted to our notion of systems.

Let $\mathcal{C}$ and $\mathcal{C}'$ be two cryptosystems, and consider the configuration depicted in Fig. 1(b), where $\mathcal{E}(\cdot,\cdot)$ is a random system with binary output, called *environment*.

Definition 1. *$\mathcal{C}$ is said to be* at least as secure as *$\mathcal{C}'$, denoted $\mathcal{C} \succ \mathcal{C}'$, if for all environments $\mathcal{E}$ the following holds: For any attacker $\mathcal{A}$ accessing $\mathcal{C}$ there is another attacker $\mathcal{A}'$ accessing $\mathcal{C}'$ such that the difference between the probability distributions of the binary outputs of $\mathcal{E}(\mathcal{C}^{\mathrm{priv}}, \mathcal{A}(\mathcal{C}^{\mathrm{pub}}))$ and $\mathcal{E}(\mathcal{C}'^{\mathrm{priv}}, \mathcal{A}'(\mathcal{C}'^{\mathrm{pub}}))$,*

$$\left|\mathrm{Prob}[\mathcal{E}(\mathcal{C}^{\mathrm{priv}}, \mathcal{A}(\mathcal{C}^{\mathrm{pub}})) = 1] - \mathrm{Prob}[\mathcal{E}(\mathcal{C}'^{\mathrm{priv}}, \mathcal{A}'(\mathcal{C}'^{\mathrm{pub}})) = 1]\right|,$$

is negligible in the security parameter k.

Similarly, $\mathcal{C}$ is computationally at least as secure as *$\mathcal{C}'$, denoted $\mathcal{C} \succcurlyeq_{c} \mathcal{C}'$, if, additionally, $\mathcal{E}$, $\mathcal{A}$, and $\mathcal{A}'$ are efficient algorithms.*

4 Indifferentiability

4.1 The Conventional Notion of Indistinguishability

Before introducing indifferentiability as a generalization of indistinguishability, we first recall the standard definition of indistinguishability. Let $\mathcal{S} = (\mathcal{S}_k)_{k\in\mathbb{N}}$ and $\mathcal{T} = (\mathcal{T}_k)_{k\in\mathbb{N}}$ be two $(\mathcal{X},\mathcal{Y})$-systems.

Definition 2. *$\mathcal{S}$ and $\mathcal{T}$ are* (computationally) indistinguishable *if for any (computationally efficient) algorithm $\mathcal{D}$ (called* distinguisher*), interacting with one of these systems and generating a binary output (0 or 1), the advantage*

$$\left|\mathrm{Prob}[D(\mathcal{S}_k)=1] - \mathrm{Prob}[D(\mathcal{T}_k)=1]\right|$$

is negligible in the security parameter k.

The relation between indistinguishability and the security of cryptosystems is summarized by the following proposition, which in its generalized form (Theorem 1) will be proven below. Let $\mathcal{S}$ and $\mathcal{T}$ be two resources which have only private interfaces.

Proposition 2. *If and only if $\mathcal{S}$ and $\mathcal{T}$ are indistinguishable, then, for every cryptosystem $\mathcal{C}(\mathcal{T})$ using $\mathcal{T}$ as a component, the cryptosystem $\mathcal{C}(\mathcal{S})$ obtained from $\mathcal{C}(\mathcal{T})$ by replacing the component $\mathcal{T}$ by $\mathcal{S}$ is at least as secure as $\mathcal{C}(\mathcal{T})$.*

The first implication, stating that the security of $\mathcal{C}(\mathcal{S})$ is an immediate consequence of the indistinguishability between $\mathcal{S}$ and $\mathcal{T}$ (and the security of $\mathcal{C}(\mathcal{T})$), is well-known in cryptography. On the other hand, to our knowledge, the (simple) observation that this condition is also necessary in general has not previously been stated explicitly.

It is important to note that Proposition 2 only applies to settings where the resources have no public interfaces, i.e., a possible opponent has no direct access to any additional information correlated with the behavior of the systems.

4.2 Generalization to Indifferentiability

We will now extend the definition of indistinguishability to resources (with private and public interfaces, as defined in Section 3). A first attempt might be to consider a distinguisher $\mathcal{D}$ accessing both the private as well as the public interfaces of the resources. However, it turns out that such an approach leads to a too strong notion of indistinguishability (with respect to Proposition 2). This means, for instance, that there are resources $\mathcal{S}$ and $\mathcal{T}$ which are not indistinguishable (according to such a definition) while, for any cryptosystem $\mathcal{C}(\mathcal{T})$ based on $\mathcal{T}$, replacing $\mathcal{T}$ by $\mathcal{S}$ has no impact on its security, i.e., the second implication of Proposition 2 would not hold.

A notion of indistinguishability overcoming this problem is formalized by the following definition, which, unlike the conventional definition, is not symmetric. Let $\mathcal{S} = (\mathcal{S}_k)_{k\in\mathbb{N}}$ and $\mathcal{T} = (\mathcal{T}_k)_{k\in\mathbb{N}}$ be two resources and let $\mathcal{D}(\mathcal{S}_k^{\mathrm{priv}}, \mathcal{S}_k^{\mathrm{pub}})$ and $\mathcal{D}(\mathcal{T}_k^{\mathrm{priv}}, \mathcal{P}(\mathcal{T}_k^{\mathrm{pub}}))$ denote the configurations of systems as depicted by Fig. 2 (a) and (b), respectively.

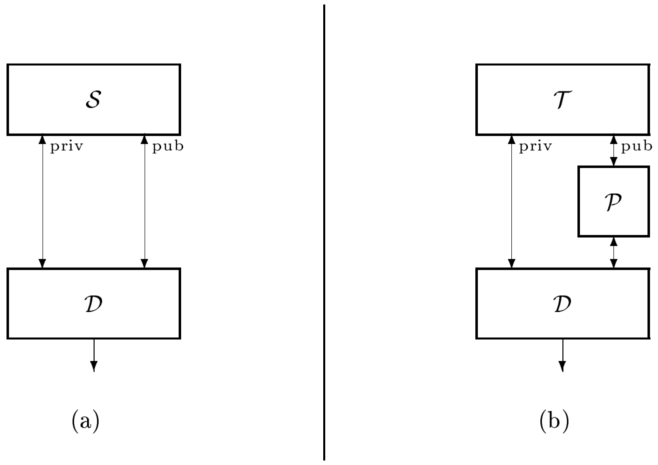


Fig. 2. Indifferentiability: The distinguisher $\mathcal{D}$ for differentiating $\mathcal{S}$ from $\mathcal{T}$ is either connected to the system $\mathcal{S}$ or the system $\mathcal{T}$. In the first case (a), $\mathcal{D}$ has direct access to the private and the public interfaces of $\mathcal{S}$, while in the latter case (b) the access to the public interfaces of $\mathcal{T}$ is replaced by an arbitrary intermediate system $\mathcal{P}$.

Definition 3. *$\mathcal{S}$ is* indifferentiable *from $\mathcal{T}$, denoted $\mathcal{S} \sqsubset \mathcal{T}$, if for any system $\mathcal{D}$ (called* distinguisher*) with binary output (0 or 1) there is a system $\mathcal{P}$ such that the advantage*

$$\left|\mathrm{Prob}[\mathcal{D}(\mathcal{S}_k^{\mathrm{priv}}, \mathcal{S}_k^{\mathrm{pub}}) = 1] - \mathrm{Prob}[\mathcal{D}(\mathcal{T}_k^{\mathrm{priv}}, \mathcal{P}(\mathcal{T}_k^{\mathrm{pub}})) = 1]\right|$$

is negligible in the security parameter k. The indifferentiability is computational*, denoted $\mathcal{S} \sqsubset_c \mathcal{T}$, if only computationally efficient algorithms are considered for $\mathcal{D}$ and $\mathcal{P}$.*

Note that indistinguishability is a special (symmetric) case of indifferentiability. Indeed, if the resources have no public interfaces, indifferentiability (Definition 3) is obviously equivalent to indistinguishability (Definition 2).

One important point about our generalization of indistinguishability is that a similar relation between the security of cryptosystems and the indifferentiability of its components as the one stated in Proposition 2 (for indistinguishability) holds. The following theorem shows that indifferentiability is the exact (i.e., necessary and sufficient) criterion needed to make general statements about the security of cryptosystems when substituting their components.

Let $\mathcal{S} = (\mathcal{S}_k)_{k \in \mathbb{N}}$ and $\mathcal{T} = (\mathcal{T}_k)_{k \in \mathbb{N}}$ be two resources.

Theorem 1. *Let $\mathcal{C}$ range over the set of all cryptosystems. Then,*

$$\mathcal{S} \sqsubset \mathcal{T} \quad \Longleftrightarrow \quad \forall\, \mathcal{C} : \mathcal{C}(\mathcal{S}) \succ \mathcal{C}(\mathcal{T}).$$

In the computational case, the same equivalence holds when "$\sqsubset$" and "$\succ$" are replaced by "$\sqsubset_c$" and "$\succ_c$", respectively.

The theorem implies that if $\mathcal{S}$ is indifferentiable from $\mathcal{T}$ and if a cryptosystem $\mathcal{C}(\mathcal{T})$ based on $\mathcal{T}$ is secure, then so is $\mathcal{C}(\mathcal{S})$, the cryptosystem obtained from $\mathcal{C}(\mathcal{T})$ by replacing the component $\mathcal{T}$ by $\mathcal{S}$. Note that the asymmetry of indifferentiability implies that there is an asymmetry on the right hand side of the equivalence in Theorem 1. In fact, even if security of $\mathcal{C}(\mathcal{S})$ implies security of $\mathcal{C}(\mathcal{T})$, then security of $\mathcal{C}(\mathcal{T})$ does not necessarily imply security of $\mathcal{C}(\mathcal{S})$.

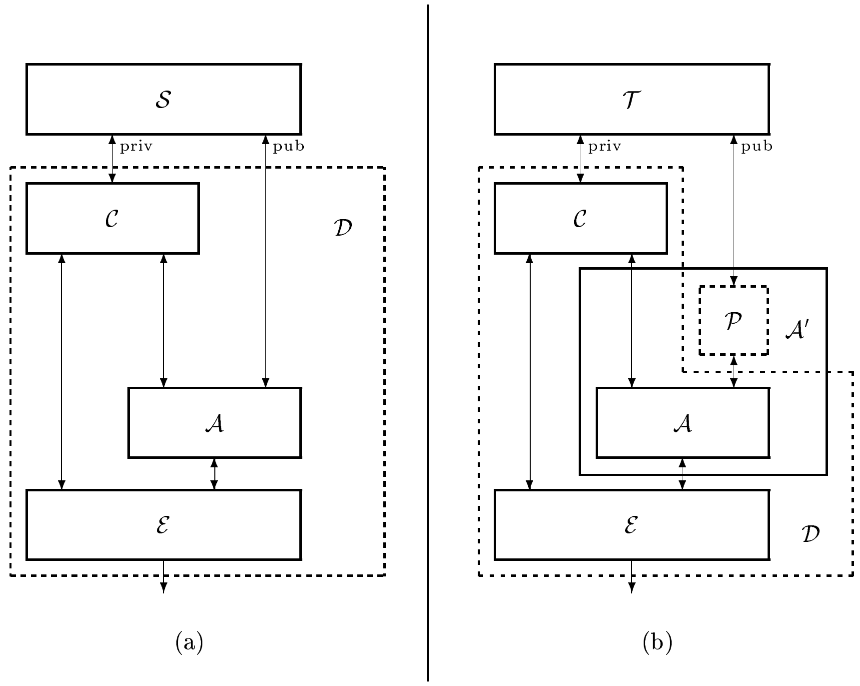


Fig. 3. Illustration for proof of Theorem 1 ("$\Longrightarrow$").

Proof. The proof is given for the information-theoretic case, where all systems might be computationally unbounded. It can however easily be adapted to hold for the computational case. To simplify the notation, set

$$d_{\mathcal{D},\mathcal{P}}(k) := \left|\text{Prob}[\mathcal{D}(\mathcal{S}_k^{\text{priv}}, \mathcal{S}_k^{\text{pub}}) = 1] - \text{Prob}[\mathcal{D}(\mathcal{T}_k^{\text{priv}}, \mathcal{P}(\mathcal{T}_k^{\text{pub}})) = 1]\right|$$

where $\mathcal{D}$ is a distinguisher, $\mathcal{P}$ an additional system, and where the configurations of systems are specified by Fig. 2 (as in Definition 3). Similarly, define

$$e_{\mathcal{E},\mathcal{C},\mathcal{A},\mathcal{A}'}(k) := \left|\text{Prob}[\mathcal{E}(\mathcal{C}(\mathcal{S}_k^{\text{priv}}), \mathcal{A}(\mathcal{S}_k^{\text{pub}})) = 1] - \text{Prob}[\mathcal{E}(\mathcal{C}(\mathcal{T}_k^{\text{priv}}), \mathcal{A}'(\mathcal{T}_k^{\text{pub}})) = 1]\right|$$

where $\mathcal{E}$ is an environment, $\mathcal{C}$ a cryptosystem, and where $\mathcal{A}$, $\mathcal{A}'$ are attackers interacting with $\mathcal{S}$ and $\mathcal{T}$, respectively (as shown in Fig. 3). The statement of

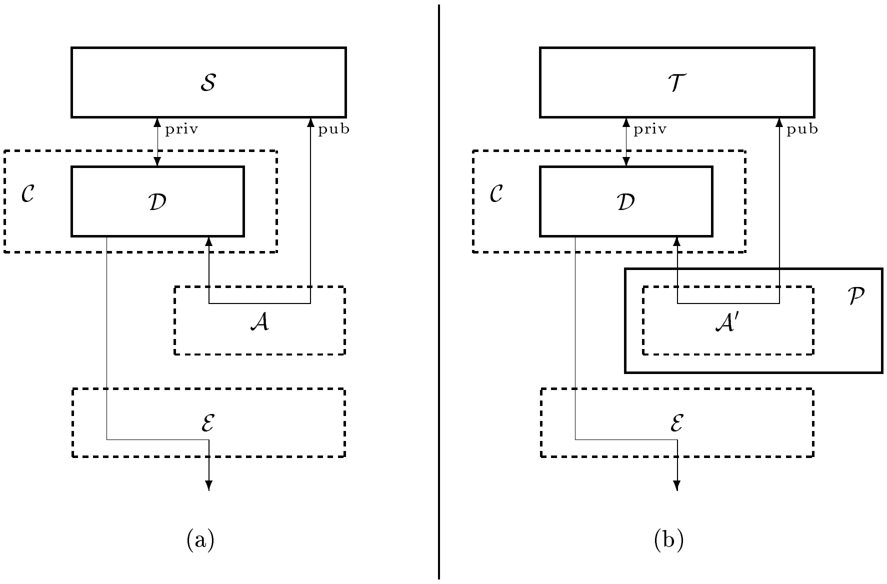


Fig. 4. Illustration for proof of Theorem 1 ("$\Longleftarrow$").

the theorem can then be rewritten as

$$\forall \mathcal{D} : \exists \mathcal{P} : d_{\mathcal{D},\mathcal{P}}(k) \text{ is negl.} \quad \Longleftrightarrow \quad \forall \mathcal{C} : \forall \mathcal{E} : \forall \mathcal{A} : \exists \mathcal{A}' : e_{\mathcal{E},\mathcal{C},\mathcal{A},\mathcal{A}'}(k) \text{ is negl.}$$

The idea for the proof is to relate both sides of this equivalence relation such that $d_{\mathcal{D},\mathcal{P}}(k) = e_{\mathcal{E},\mathcal{C},\mathcal{A},\mathcal{A}'}(k)$ holds.

Let us start with the first implication ("$\Longrightarrow$"). Let $\mathcal{C}$ be any cryptosystem, $\mathcal{E}$ an environment and $\mathcal{A}$ an attacker. Define the distinguisher $\mathcal{D}$ as the system resulting from $\mathcal{C}$, $\mathcal{E}$, and $\mathcal{A}$ being combined as shown in Fig. 3(a), and let $\mathcal{P}$ be the system such that $d_{\mathcal{D},\mathcal{P}}(k)$ is negligible in k. Finally, define the attacker $\mathcal{A}'$ as $\mathcal{A}(\mathcal{P})$ (cf. Fig. 3(b)). The two settings involving the system $\mathcal{S}$ (represented in Fig. 3(a) by solid lines and dashed lines, respectively) as well as the two settings involving the system $\mathcal{T}$ (Fig. 3(b)) are then obviously equivalent, i.e., the probabilities of their outputs are equal. Consequently, $e_{\mathcal{E},\mathcal{C},\mathcal{A},\mathcal{A}'}(k)$ equals $d_{\mathcal{D},\mathcal{P}}(k)$, i.e., $e_{\mathcal{E},\mathcal{C},\mathcal{A},\mathcal{A}'}(k)$ is negligible.

The second implication ("$\Longleftarrow$") is proven similarly. Let $\mathcal{D}$ be any distinguisher. Let the cryptosystem $\mathcal{C}$ be identical to $\mathcal{D}$,[6] and define the environment $\mathcal{E}$ and the attacker $\mathcal{A}$ as a trivial system simply forwarding all queries as shown in Fig. 4(a). Let $\mathcal{A}'$ be an attacker such that $e_{\mathcal{E},\mathcal{C},\mathcal{A},\mathcal{A}'}(k)$ is negligible in k. Finally,

[6] Motivated by a construction given in [6], one could also define a more "realistic" cryptosystem containing $\mathcal{D}$ such that, if $\mathcal{D}$ outputs 0, it performs some useful task, while, if $\mathcal{D}$ outputs 1, it behaves completely insecurely by revealing some secret information.

define $\mathcal{P} := \mathcal{A}'$ (cf. Fig. 4(b)). Again, the two settings involving the system $\mathcal{S}$ (Fig. 4(a)) as well as the two settings involving the system $\mathcal{T}$ (Fig. 4(b)) are equivalent, i.e., $d_{\mathcal{D},\mathcal{P}}(k)$ equals $e_{\mathcal{E},\mathcal{C},\mathcal{A},\mathcal{A}'}(k)$ and is thus negligible. $\square$

5 Reductions and Reducibility

In cryptography one often asks whether a given system $\mathcal{V}$ can be used to construct a (seemingly stronger) system $\mathcal{U}$ which is specified by its functionality. If this is the case, one says that $\mathcal{U}$ is *reducible* to $\mathcal{V}$. The formal definition of reducibility makes clear that this concept is strongly related to the notion of indistinguishability, or, in our generalized setting, to indifferentiability.

Let $\mathcal{U}$ and $\mathcal{V}$ be two resources.

Definition 4. *$\mathcal{U}$ is* information-theoretically securely (computationally securely) reducible *to $\mathcal{V}$, denoted $\mathcal{U} \to \mathcal{V}$ ($\mathcal{U} \underset{c}{\to} \mathcal{V}$), if there exists a (computationally efficient) algorithm $\mathcal{B} \in \Gamma^{\mathrm{p}}(\mathcal{V}/\mathcal{U})$ such that $\mathcal{B}(\mathcal{V}) \sqsubset \mathcal{U}$ ($\mathcal{B}(\mathcal{V}) \underset{c}{\sqsubset} \mathcal{U}$).*

Analogously to indistinguishability and indifferentiability, the concept of reducibility is useful for cryptographic security proofs. The following theorem is a direct consequence of Theorem 1 and the above definition of reducibility.

Theorem 2. *Let $\mathcal{C}$ range over the set of all cryptosystems. Then,*

$$\mathcal{U} \to \mathcal{V} \quad \Longleftrightarrow \quad \exists \mathcal{B} \in \Gamma^{\mathrm{p}}(\mathcal{V}/\mathcal{U}) : \forall \mathcal{C} : \mathcal{C}(\mathcal{B}(\mathcal{V})) \succ \mathcal{C}(\mathcal{U}).$$

In the computational case, the same statement holds when "$\to$" and "$\succ$" are replaced by "$\underset{c}{\to}$" and "$\underset{c}{\succ}$", respectively.

6 A Sufficient Criterion for Irreducibility

The following theorem gives an easily verifiable sufficient criterion for a public resource $\mathcal{U}$ not to be reducible to another public resource $\mathcal{V}$. This criterion will be formulated in terms of the entropy of the output generated by these resources, as defined in Section 3.

Let $\mathcal{U} = (\mathcal{U}_k)_{k\in\mathbb{N}}$ and $\mathcal{V} = (\mathcal{V}_k)_{k\in\mathbb{N}}$ be two public resources with costs given by $c_{\mathcal{U}_k}$ and $c_{\mathcal{V}_k}$, respectively. For convenience, let us assume that for fixed t, the entropies $h^{\infty}_{\mathcal{U}_k}(t)$ and $h^{0}_{\mathcal{V}_k}(t)$ are monotonically increasing in k. Informally speaking, the theorem states that $\mathcal{U}$ is not reducible to $\mathcal{V}$ if $h^{\infty}_{\mathcal{U}_k}(t)$ grows "sufficiently faster than" $h^{0}_{\mathcal{V}_k}(t)$.

Theorem 3. *If for each $k \in \mathbb{N}$ and any polynomial p the function $h^{\infty}_{\mathcal{U}_k}$ grows asymptotically faster than the function $h^{0}_{\mathcal{V}_k} \circ p$, then $\mathcal{U} \not\to \mathcal{V}$.*

A similar theorem holds for the computational case. (In the proof given below, the main changes needed to obtain a computational version are indicated.) The proof mainly follows the lines of the proof of Proposition 1 given in Section 2:

It is shown that for any reduction $\mathcal{B}(\cdot)$, there exists a distinguisher for differentiating $\mathcal{B}(\mathcal{V})$ from $\mathcal{U}$. The idea is to let the distinguisher simulate $\mathcal{B}(\mathcal{V})$ and then check whether this simulation corresponds to the behavior of the resource it is connected to ($\mathcal{U}$ or $\mathcal{B}(\mathcal{V})$). By an entropy argument, it can be concluded that this test fails (with high probability) if (and only if) the distinguisher is connected to $\mathcal{U}$.

Proof. It has to be shown that $\mathcal{B}(\mathcal{V}) \not\sqsubset \mathcal{U}$ for any $\mathcal{B}(\cdot) \in \Gamma^{\mathrm{p}}(\mathcal{V}/\mathcal{U})$. By the definition of $\Gamma^{\mathrm{p}}(\mathcal{V}/\mathcal{U})$, $\mathcal{B}(\mathcal{V})$'s costs $\bar{c}_k$ are bounded by a polynomial p in the costs $c_{\mathcal{U}_k}$ of $\mathcal{U}_k$ and the security parameter k,

$$\bar{c}_k(x) \leq p_k(c_{\mathcal{U}_k}(x)) \ . \tag{1}$$

Similarly to the proof presented in Section 2, we first give an explicit construction of a distinguisher for differentiating $\mathcal{B}(\mathcal{V})$ from $\mathcal{U}$, and then show that it has all the desired properties.

Construction of $\mathcal{D}$ The distinguisher $\mathcal{D}(\cdot,\cdot)$ for differentiating $\mathcal{B}(\mathcal{V})$ from $\mathcal{U}$ has two interfaces (cf. Fig. 2 where $\mathcal{S} = \mathcal{B}(\mathcal{V})$ and $\mathcal{T} = \mathcal{U}$) which we call $\mathcal{D}^{\mathrm{priv}}$ and $\mathcal{D}^{\mathrm{pub}}$, respectively.

For $r \in \mathbb{N}$, let the min-entropy $H_\infty(Y_1 \cdots Y_r)$ of all outputs Y_i of the system $\mathcal{U}_k$ on inputs $x_i := i$ (for $i = 1, \ldots, r$) be denoted as $\bar{h}_k(r)$, and let l be some positive integer to be determined later. For simplicity, let us assume (without loss of generality) that the functions $\bar{h}_k$ as well as $h^\infty_{\mathcal{U}_k}$ are invertible, and that the outputs of $\mathcal{V}$ are single bits.

$\mathcal{D}$ is constructed as follows: First, $\mathcal{D}$ sends queries $x'_j := j$ for $j = 1, \ldots, l$ to interface $\mathcal{D}^{\mathrm{pub}}$ and stores the received answers $z_1, \ldots, z_l$ (which by assumption are single bits). Then, $\mathcal{D}$ subsequently simulates $\mathcal{B}(\mathcal{V})$ on test inputs $x_i := i$ for $i = 1, \ldots, n$ where $n := (\bar{h}_k)^{-1}(l+k)$, resulting in outcomes $\bar{y}_i$. For the simulation of $\mathcal{B}$, any query $x' \in \{1, \ldots, l\}$ of $\mathcal{B}$ to $\mathcal{V}$ is answered by the corresponding stored value $z_{x'}$. If $x' > l$, $\mathcal{D}$ stops with output 0. The same test inputs x_i are then sent to interface $\mathcal{D}^{\mathrm{priv}}$, resulting in answers y_i. If $y_i = \bar{y}_i$ for all $i = 1, \ldots, n$, $\mathcal{D}$ outputs 1, and 0 otherwise.

The above construction of $\mathcal{D}$ must be modified slightly in order to avoid the following technical problem: The stored values $z_1, \ldots, z_l$ might be arbitrarily chosen by $\mathcal{P}$, in which case they do not necessarily correspond to (potential) outputs of $\mathcal{V}$. The number of queries of the simulated system $\mathcal{B}$ and, in the computational case, the running time of the simulation of $\mathcal{B}$, might thus be unbounded when using $z_1, \ldots, z_l$ as answers for simulating $\mathcal{B}$'s queries. To overcome this problem, $\mathcal{D}$ simply stops the simulation of $\mathcal{B}$ on input x after some maximal number $t_{\max}(x)$ of queries (and, in the computational case, some maximal number $t'_{\max}(x)$ of computational steps) of $\mathcal{B}$, where $t_{\max}(x)$ (and $t'_{\max}(x)$) is the maximal number of queries (computational steps) of $\mathcal{B}$ when receiving correct answers to its queries.

It remains to show that $\mathcal{D}$ satisfies the following properties:

(a) $\mathcal{D}(\mathcal{U}^{\mathrm{priv}}, \mathcal{P}(\mathcal{U}^{\mathrm{pub}}))$ outputs 1 with negligible probability in k.
(b) $\mathcal{D}(\mathcal{B}(\mathcal{V}^{\mathrm{priv}}), \mathcal{V}^{\mathrm{pub}})$ outputs 1 with certainty.

$\mathcal{D}$ satisfies property (a). Note that $\mathcal{D}$ can only have output 1 if the n-tuples $y = (y_1, \ldots, y_n)$ and $\bar{y} = (\bar{y_1}, \ldots, \bar{y_n})$ are equal. It thus suffices to verify that the probability of this event is negligible in k.

Since $\bar{y}$ is fully specified by the bits $z_1, \ldots, z_l$ used for the simulation of $\mathcal{B}(\mathcal{V})$ (note that $\mathcal{B}$ is deterministic) there are at most 2^l possible values for $\bar{y}$. Let $\bar{\mathcal{Y}}$ be the set of these 2^l values. Obviously, y can only be equal to $\bar{Y}$ if $y \in \bar{\mathcal{Y}}$. This happens with probability at most

$$\sum_{y \in \bar{\mathcal{Y}}} P_Y(y) \leq |\bar{\mathcal{Y}}| \cdot \max_{y \in \bar{\mathcal{Y}}} P_Y(y) \leq 2^l \cdot 2^{-H_\infty(Y)} = 2^l \cdot 2^{-\bar{h}_k(n)} \leq 2^{-k} ,$$

which concludes the proof of property (a).

$\mathcal{D}$ satisfies property (b). We first show that the property holds for l satisfying

$$l \geq h^0_{\mathcal{V}_k}(p_k((h^\infty_{\mathcal{U}_k})^{-1}(l+k))), \tag{2}$$

where $p_k(\cdot)$ is defined as in (1). Second, we prove that condition (2) is always satisfied for l large enough (but polynomially bounded in the computational case).

By the definition of $h^\infty_{\mathcal{U}_k}$, $c_{\mathcal{U}_k}(x) \leq (h^\infty_{\mathcal{U}_k})^{-1}(l+k)$ holds for all queries $x = 1, \ldots, n$. By assumption, the costs $c_{\mathcal{U}}$ and $\bar{c}$ (of $\mathcal{U}$ and $\mathcal{B}(\mathcal{V})$, respectively) satisfy condition (1). The costs $c_{\mathcal{V}_k}(x')$ of $\mathcal{V}_k$ for each potential query x' of $\mathcal{B}$ to $\mathcal{V}$ are thus bounded by

$$c_{\mathcal{V}_k}(x') \leq p_k((h^\infty_{\mathcal{U}_k})^{-1}(l+k)) .$$

Let $x_{\max}$ be the maximal query of $\mathcal{B}$ to $\mathcal{V}$ (i.e., $x' \leq x_{\max}$ for all queries of $\mathcal{B}$). It follows from the definition of h^0 that the length l' of the list containing $\mathcal{V}$'s answers to the queries $1, \ldots, x_{\max}$ satisfies

$$l' \leq h^0_{\mathcal{V}_k}(p_k((h^\infty_{\mathcal{U}_k})^{-1}(l+k))) .$$

By construction, $\mathcal{D}$ outputs 1 if the list of stored values $z_1, \ldots, z_l$ contains the (correct) answers to all queries x' of $\mathcal{B}$ to $\mathcal{V}^{\text{priv}}$ (note that, by assumption, $\mathcal{B}$ is deterministic). Clearly, this is the case if $l' \leq l$, which is true if l satisfies inequality (2).

It remains to prove that (2) holds for l large enough: By assumption, for any $k \in \mathbb{N}$, the function $h^0_{\mathcal{V}_k} \circ p_k \circ (h^\infty_{\mathcal{U}_k})^{-1}$ grows slower than the identity function. Hence

$$\lim_{l \to \infty} \frac{l}{h^0_{\mathcal{V}_k}(p_k((h^\infty_{\mathcal{U}_k})^{-1}(l+k)))} \geq 1 ,$$

which implies that (for any fixed k) there is a value for l satisfying (2).

7 Applications

7.1 Random Oracles, Asynchronous Beacons, and Finite Random Strings

We will now apply the framework presented in the previous sections to prove separation results for random oracles, beacons, and finite random strings. Each of these cryptographic primitives can be modeled as a public resource $\mathcal{S}$ whose outputs only depend on the previous inputs (i.e., $\mathcal{S}$ is a random function, providing identical private and public interfaces with input set $\mathcal{X} = \mathbb{N}$ and output set $\mathcal{Y} = \{0, 1\}$).[7] Each query $x \in \mathcal{X}$ to $\mathcal{S}$ is answered by R_x where $R = R_1 R_2 \cdots$ is a (possibly infinite) bitstring randomly chosen according to some distribution P_R.

Random oracles, beacons, and finite random strings only differ by the length of the string R and the cost function c. For a *random oracle* $\mathcal{R}$, R has infinite length and the costs are $c(x) := 1$, or, alternatively, $c(x) := |x|$, where $|x|$ denotes the length of x. (In the following, we only need an upper bound for the costs of a random oracle, i.e., we will assume that $c(x) \leq |x|$.) For an *asynchronous beacon* $\mathcal{Q}$, R is also an infinite bitstring, but the costs for the queries are higher, namely $c(x) := x$. On the other hand, for a *finite random string* $\mathcal{F}$, the length $|R|$ of R is given as a function in the security parameter k which is bounded by a polynomial p, and the costs are $c(x) := C$ for some constant C. Moreover, for any query on input x with $x > |R|$ the output is 0. In the following, we say that a random oracle, beacon, or finite random string is *uniform* if R is uniformly distributed, and denote these objects as $\overline{\mathcal{R}}$, $\overline{\mathcal{Q}}$, and $\overline{\mathcal{F}}$, respectively.

7.2 Impossibility Results

It is obvious that an asynchronous beacon can always be reduced to a random oracle (using an algorithm which merely passes on the inputs and outputs) and that a finite random string can always be reduced to a beacon (using the same trivial algorithm which additionally checks that the input is not larger than some predefined bound). The inverse reductions are, however, not possible.

Theorem 4. *The following irreducibility results hold for both the information-theoretic and the computational case (where "$\not\twoheadrightarrow$" is replaced by "$\not\hookrightarrow$"):*

$$\overline{\mathcal{R}} \not\twoheadrightarrow \mathcal{Q} \qquad \textit{and} \quad \overline{\mathcal{Q}} \not\twoheadrightarrow \mathcal{F}.$$

Proof. The main task required for the proof of this theorem is the computation of the entropies according to the definitions in Section 3. The assertion then follows directly from Theorem 3. For a random oracle, we obtain

$$h^{\infty}_{\overline{\mathcal{R}}_k}(t) = h^{0}_{\mathcal{R}_k}(t) \geq \sum_{i=1}^{t} 2^i = 2^{t+1} - 2,$$

and similarly, for an asynchronous beacon,

$$h^{\infty}_{\overline{\mathcal{Q}}_k}(t) = h^{0}_{\mathcal{Q}_k}(t) = t$$

[7] We will assume that the outputs of random oracles, beacons and finite random strings are single bits. This entails no restriction of generality since any of these random functions providing outputs of some length l can efficiently be reduced to a corresponding random function with outputs of length 1 (as long as l grows only polynomially in the security parameter k).

(independently of $k \in \mathbb{N}$). Since for a finite random string the length of R is given by a function in the security parameter k which is bounded by a polynomial p in k, we have

$$h^{\infty}_{\overline{\mathcal{F}}_k}(t) = h^0_{\mathcal{F}_k}(t) \leq \begin{cases} 0 & \text{if } t < C \\ p(k) & \text{otherwise.} \end{cases}$$

(for all $k \in \mathbb{N}$). Note that the above expressions for $h^0_{\mathcal{R}_k}$, $h^0_{\mathcal{Q}_k}$ and $h^0_{\mathcal{F}_k}$ also hold if the respective systems are not uniform. $\square$

Together with Theorem 2, one can conclude that a random oracle in general can not be replaced by any algorithm interacting with an asynchronous beacon, and similarly, a beacon can not be replaced by any algorithm interacting with a public finite random string without affecting the security of an underlying cryptosystem. The failure of the random oracle methodology can thus be seen as a direct consequence of each of the two irreducibility results of Theorem 4.

8 Conclusions

One crucial motivation for introducing the notion of indifferentiability is that it characterizes exactly when one can replace a subsystem of a cryptosystem by another subsystem without affecting the security. In contrast to indistinguishability, indifferentiability is applicable in the important case of settings where a possible adversary is assumed to have access to additional information about a system. This generality is for instance crucial in the setting of the random oracle methodology, and our abstract framework yields as a simple consequence, actually of each of two different impossibility results, the impossibility result by Canetti, Goldreich and Halevi [6] stating that random oracles can not be implemented. In view of the highly involved arguments of [6] based on CS-proofs, we hope to have presented a more generic approach to arguing about such impossibility results, thus also applicable in other contexts where systems have public parameters or where an adversary can obtain side-information about secret parameters.

References

1. M. Bellare, A. Boldyreva, and A. Palacio. An un-instantiable random-oracle-model scheme for a hybrid-encryption problem. ePrint archive: `http://eprint.iacr.org/2003/077/`, 2003.
2. M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In V. Ashby, editor, *1st ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.
3. 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.
4. R. Canetti. Security and composition of multi-party cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

5. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 136–145, 2001.
6. R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing*, pages 209–218. ACM Press, 1998.
7. R. Canetti, O. Goldreich, and S. Halevi. On the random-oracle methodology as applied to length-restricted signature schemes. ePrint archive: `http://eprint.iacr.org/2003/150/`, 2003.
8. 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.
9. 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.
10. U. Maurer. Indistinguishability of random systems. In *Advances in Cryptology — EUROCRYPT '02*, volume 2332 of *Lecture Notes in Computer Science*, pages 110–132. Springer-Verlag, 2002.
11. S. Micali. CS proofs. In *Proc. 35th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 436–453. IEEE, 1994.
12. J. B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: the non-committing encryption case. In *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 111–126. Springer-Verlag, 2002.
13. 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.
14. B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In *7th ACM Conference on Computer and Communications Security*, pages 245–254. ACM Press, 2000.
15. B. Pfitzmann and M. Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *IEEE Symposium on Security and Privacy*, pages 184–200. IEEE Computer Society Press, 2001.
16. 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.
17. C. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.