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

Ueli Maurer Renato Renner Clemens Holenstein

Department of Computer Science
Swiss Federal Institute of Technology (ETH), Zurich
CH-8092 Zurich, Switzerland
{maurer,renner,holenstein}@inf.ethz.ch

Abstract. The goals of this paper are three-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.
Second, we prove that indifferentiability is the necessary and sufficient condition on two systems $\mathcal{S}$ and $\mathcal{T}$ such that the security of any cryptosystem using $\mathcal{T}$ as a component is not affected when $\mathcal{T}$ is substituted by $\mathcal{S}$. In contrast to 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.
Third, 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.

Key words. Indistinguishability, reductions, indifferentiability, security proofs, random oracle methodology, hash functions.

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.

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.

[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 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 [5]. 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 consider 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).

An example for a system, which we will consider more closely, is the random oracle. Its importance in cryptography is due to the so called random oracle methodology, first made explicit by Bellare and Rogaway [1], where the security of cryptosystems is proven under the assumption that any party has access to a random oracle. The methodology has later been used in many papers (e.g. [7, 8, 15, 11, 1, 10, 2, 14]). A (binary) random oracle $\mathcal{R}$ can be thought of as an infinite sequence $R_1, R_2, \ldots$ of random bits where the nth bit R_n can be accessed in constant time.

We also introduce a slightly weaker primitive, called (binary) asynchronous beacon[2] $\mathcal{Q}$, defined as a sequence of of random bits $R_1, R_2, \ldots$ which is only sequentially accessible, i.e., the time needed to access R_n is $O(n)$. Thus the only difference between a random oracle and a beacon is the cost associated with accessing the randomness. 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.)

One can also consider weaker variants of random oracles or beacons whose bits are not uniformly random, not independent, or both. Moreover, one can consider systems between an asynchronous beacon and a random oracle for which one can access the bits $R_1, R_2, \ldots$ faster than sequentially but not in an arbitrary (random access) manner.[3] Another system of interest is a finite random string $\mathcal{F}$ which can be assumed to be given completely at unit cost $O(1)$. In a sense, a random oracle and a finite random string are two extreme points on the scale of systems we consider, 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

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

[3] For instance, the cost of accessing R_n could be $O((\log n)^k)$ for some k, or $O(n^\alpha)$ for some $\alpha < 1$, or any other function of n.

to the above question is characterized by the rates at which entropy can be accessed in the systems $\mathcal{U}$ and $\mathcal{V}$ (Section 6). 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 (Section 7). This also proves the main result of [5] 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 $\mathcal{C}(\mathcal{U})$.

Allowing for this general type of security proofs, the indistinguishability of two systems seems to be a strong property. Nevertheless, it is in a certain sense the weakest possible requirement needed for security proofs of this type. In fact, if two components $\mathcal{S}$ and $\mathcal{T}$ are not indistinguishable, then there exists a cryptosystem $\mathcal{C}(\mathcal{T})$ based on $\mathcal{T}$ which is secure, while the system $\mathcal{C}(\mathcal{S})$ constructed from $\mathcal{C}(\mathcal{T})$ by replacing $\mathcal{T}$ by $\mathcal{S}$ is insecure.

However, these considerations are all subject to the assumption that each component (or *primitive*) a cryptosystem is based on is a resource belonging to one specific party which has exclusive access to it, i.e., all other entities are unable to directly influence the component's 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* (Section 4). Together with its derived notion of reducibility (Section 5), 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.

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

As a motivating example for the general problem considered in this paper, we give a straightforward proof of the classical impossibility result by Canetti, Goldreich, and Halevi [5] that a random oracle cannot be realized by a (family of) hash functions, showing intrinsic limitations of the random oracle methodology. The original proof is quite involved as it is based on techniques like Micali's CS-proofs [10]. Very recently, the same authors proposed a new proof [6] of their result based on a new type of interactive proof systems.

While this impossibility result also follows directly from our general impossibility results (cf. Section 7), we give a self-contained proof in this section before introducing the definitions required to formalize the general problem. After seeing this proof, it will be easier for the reader to understand the motivation for the definitions and to follow the rest of the paper.

We show the following proposition which directly implies the separation result as formulated in [5]:

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.*
- *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 proposition is proven by an explicit construction of $\mathcal{C}$ based on an algorithm $\mathcal{D}(\cdot)$ such that the behavior of $\mathcal{D}(\mathcal{R})$ is different from $\mathcal{D}(f)$. More precisely, let $\mathcal{D}(\cdot)$ be 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) $\mathcal{D}(\mathcal{R})$ outputs 0 for any input with overwhelming probability (over the randomness of $\mathcal{R}$).
(b) For any efficiently computable function f, there exists an input m causing $\mathcal{D}(f)$ to output 1.[4]
(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).

Construction of $\mathcal{C}$. Given an efficient signature scheme $\bar{\mathcal{C}}(\cdot)$ such that $\bar{\mathcal{C}}(\mathcal{R})$ is secure,[5] a new efficient signature scheme $\mathcal{C}(\cdot)$ is obtained 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 $\bar{\mathcal{C}}(\cdot)$). Otherwise, if $\mathcal{D}(\cdot)$'s output is 1, the signing algorithm of $\mathcal{C}(\cdot)$ behaves completely insecurely (e.g., by revealing a secret key, as proposed in [5]).

(In)security and efficiency of $\mathcal{C}$. 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 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 $\bar{\mathcal{C}}(\cdot)$.

[4] Moreover, in our construction of D, m can easily be determined given an algorithm which efficiently computes f.
[5] i.e., the success probability of any attacker against $\bar{\mathcal{C}}(\mathcal{R})$ is negligible in k.

Construction of $\mathcal{D}$. It remains to be proven that an algorithm $\mathcal{D}(\cdot)$ with the desired properties (a)–(c) indeed exists. We give an explicit construction for $\mathcal{D}(\cdot)$:

$\mathcal{D}(\cdot)$ interprets its input m as a pair (π, t) consisting of an encoding of a program π for a universal Turing machine $\mathcal{U}$ and a unary encoding of some integer t (in particular, $t \leq |m|$). Let $q = 2|\pi| + k$ (where $|\pi|$ is the length of a binary representation of π). For inputs $x = 1, \ldots, q$, $\mathcal{D}(\cdot)$ simulates at most t steps of the program π on $\mathcal{U}$, resulting in outcomes $\pi(1), \ldots, \pi(q)$.[6] 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.

Let us now show that $\mathcal{D}(\cdot)$ satisfies properties (a) – (c):

$\mathcal{D}$ 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 $\mathcal{R}$) equals the output $\pi(x)$ (of the fixed program π) with probability at most $1/2$ (assuming that each output of $\mathcal{R}$ consists of at least one random bit), 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.[7]

$\mathcal{D}$ 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 the size of π 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 $\mathcal{D}(\mathcal{R})$ is at most $O((2|\pi| + k) \cdot t) \leq O((|m| + k)^2)$. □

3 Basic Definitions and Notation

The main results in this paper are statements about the security of cryptosystems and the relations between the components they are based on. These objects can generically be described as (interacting) systems, defined by their input/output behavior (Subsection 3.1). Additionally, to analyze constructions of resources based on weaker primitives, a notion of efficiency for systems (Subsection 3.2) is required. Finally, we need a general definition of security for cryptosystems (Subsection 3.3).

[6] If the program π (run on $\mathcal{U}$) does not generate an output after t steps, $\pi(i)$ is set to some dummy value.

[7] Note that, given the program π, the maximum running time t and, consequently, the input m, can easily be computed.

3.1 Interacting Systems

Any type of (cryptographic) components or resources as well as the parties interacting with them can be characterized as systems. For their representation, we will basically adapt the terminology introduced in [9]. A $(\mathcal{X},\mathcal{Y})$-*system* is a sequence of conditional probability distributions $P_{Y_i|X^iY^{i-1}}$ (for $i \in \mathbb{N}$) where $X^i := [X_1,\ldots,X_i]$ and $Y^{i-1} := [Y_1,\ldots,Y_{i-1}]$ and 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 *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 finite bitstrings. Since there is a canonical bijection between $\mathbb{N}$ and $\{0,1\}^*$,[8] any finite bitstring can be identified with a natural number. From now on, saying that some number $i \in \mathbb{N}$ is given as input to a system $\mathcal{S}$, we rather mean its representation as a binary string.

We will consider configurations of interacting systems where each system has certain interfaces which are connected to the interfaces of other systems.[9] A *configuration of systems* is a set of systems where the systems' interfaces are pairwise connected.[10]

Any configuration of interacting systems can be seen as a new system. Let, for instance, $\mathcal{S}$ 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 $\mathcal{S}$ (cf. 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 can often be denoted similarly, e.g., $\mathcal{S}(\mathcal{U}^1,\mathcal{V}(\mathcal{U}^2))$ where $\mathcal{S}(\cdot,\cdot)$ is a system being connected to a first interface of a system $\mathcal{U}$ (denoted $\mathcal{U}^1$) as well as to a system $\mathcal{V}(\cdot)$ where the latter is itself connected to a second interface of $\mathcal{U}$ (denoted $\mathcal{U}^2$) (cf. Fig. 1(b)).

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 class $(\mathcal{S}_k)_{k\in\mathbb{N}}$ parameterized by k, where each $\mathcal{S}_k$ is a system in the sense described above. Furthermore, a function $f : k \mapsto f(k)$ is said to be *negligible* in k if $f(k)$ decreases faster than the inverse of any polynomial in k, i.e., $\lim_{k\to\infty} f(k)\cdot p(k) = 0$ for any polynomial p.

For instance, a security parameter is needed to define the computational efficiency of algorithms interacting with 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.[11]

3.2 A Notion of Efficiency for Systems

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 neces-

[8] One could take the binary representation of the natural numbers, dropping the highest bit (which always is a 1).

[9] Formally, the inputs X_i and outputs Y_i of a *system with n interfaces* are pairs, (X_i', S_i) and (Y_i', R_i), respectively, where S_i and R_i take on values in $\{1,\ldots,n\}$ indicating the interfaces to be used.

[10] If the interface with index s of a system $\mathcal{S}$ is *connected* to the interface t of a system $\mathcal{T}$, then any output of $\mathcal{S}$ of the form (m,s) (for any arbitrary message m) is given as input (m,t) to $\mathcal{T}$.

[11] We will always implicitly assume that k is given as input to any algorithm, but that the algorithms themselves do not depend on k.

sarily 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 real range 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 $\mathbb{R}^+$.

The costs of a composite system $\mathcal{U}$ consisting of r systems $\mathcal{S}^{(1)},\ldots,\mathcal{S}^{(r)}$ are defined as the sum of the costs of these systems when processing an input of $\mathcal{U}$. Consequently, if c is the cost function of a composite system $\mathcal{U} := \mathcal{S}(\mathcal{T})$ and $\bar{c}$ is the cost function of its component $\mathcal{T}$, then (for all inputs x of $\mathcal{S}(\mathcal{T})$)

$$c(x) \geq \sum_{i=1}^{n} \bar{c}(\bar{x}_i),$$

where $\bar{x}_1,\ldots,\bar{x}_n$ are $\mathcal{S}$'s queries (for some $n \in \mathbb{N}$) sent to $\mathcal{T}$ while processing x.

Similarly to the usual notion of computational efficiency of algorithms, we say that a system $\mathcal{S}$ (i.e., 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 is bounded by a polynomial in the input length and the security parameter k, i.e., if there exists a polynomial p in two arguments such that $c_k(x) \leq p(|x|,k)$ for all x in the input set of $\mathcal{S}$ and $k \in \mathbb{N}$. Additionally, for two systems $\mathcal{U}$ and $\mathcal{V}$, we define $\Gamma(\mathcal{V}/\mathcal{U})$ to be the set of all systems $\mathcal{B}$ such that the costs $\bar{c}_k$ of the system $\mathcal{B}(\mathcal{V})$ are bounded by a polynomial in the costs c_k of the system $\mathcal{U}$ and the security parameter k. Clearly, for any $\mathcal{B} \in \Gamma(\mathcal{V}/\mathcal{U})$, 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[12], 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 $H_0(X) := \log_2|\mathcal{X}|$,[13] and where H_∞ is the min-entropy (defined as $H_\infty(X) := -\log_2 \max_{x\in\mathcal{X}} P_X(x)$). Clearly, $h^0_{\mathcal{S}}$ and $h^\infty_{\mathcal{S}}$ are monotonically increasing functions, and $h^0_{\mathcal{S}}(t) \geq h^\infty_{\mathcal{S}}(t)$.

3.3 Cryptosystems and Security

In the following, we will consider settings involving n separated systems, called *parties* $\mathcal{P}_1,\ldots,\mathcal{P}_n$, having access to certain resources, i.e., systems providing interfaces to these parties. As an example, a broadcast channel for $\mathcal{P}_i$ is the system which simply takes an input value from its interface to $\mathcal{P}_i$ and sends this value to all other interfaces, i.e., to all other parties connected to the resource. Similarly, a private channel from $\mathcal{P}_i$ to $\mathcal{P}_j$ is the system sending all inputs from $\mathcal{P}_i$ as output to the interface $\mathcal{P}_j$ is connected to.

Resources can be used as building blocks for constructing new, more powerful resources. For n parties, such a construction is specified by a set of n algorithms $\mathcal{B}_1,\ldots,\mathcal{B}_n$ which are run by

[12] Similar to the inputs of a system, the inputs of a cost function can be regarded as natural numbers.

[13] This definition requires a specification of the range of the sequence $Y_1,\ldots,Y_{n_t}$. For the rest of this paper, we will assume that this range corresponds to a set of bitstrings of some fixed length l_t, such that $h^0_{\mathcal{S}_k}(t) \leq l_t$.

the parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$ and make calls to the original resources. For instance, given a broadcast channel for a party $\mathcal{P}_i$, it is easy to construct a private channel from $\mathcal{P}_i$ to any arbitrary party $\mathcal{P}_j$: All parties except $\mathcal{P}_j$ simply have to ignore the values they obtain from the broadcast resource. This, however, only works under the assumption that all parties behave correctly.

In a cryptographic setting, when making statements about a given configuration of parties interacting with resources, we generally have to distinguish between two different types of players, namely *honest* and *dishonest parties*, where the latter will be considered as being controlled by one central *adversary*.

The access of the honest parties and the adversary to a given system $\mathcal{S}$ (or, more generally, a set of systems) providing m interfaces is specified by a so called *access structure*.[14] This is a set of pairs (P, A), each of them describing a concrete constellation of honest parties and an adversary interacting with the system. P and A are disjoint subsets of $\{1, \ldots, m\}$ containing the labels of interfaces of $\mathcal{S}$ connected to honest parties and the interfaces controlled by the adversary, respectively. For instance, the setting shown in Fig. 2 for a resource $\mathcal{S}$ providing 7 interfaces is characterized by the pair $(P, A) = (\{1, 2, 3, 4\}, \{6, 7\})$.[15]

A *cryptosystem* $\mathcal{C}$ can generally be seen as a system providing interfaces to be used by certain entities. Its security is (with respect to an access structure Σ) characterized relative to an ideal system which *by definition* is secure.[16] 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 $\mathcal{C}'$. Our definition is based on ideas proposed by Canetti [3, 4], and by Pfitzmann and Waidner [12, 13] (for the case of static adversaries), adapted to our general notion of systems.

Informally, a cryptosystem $\mathcal{C}$ is said to be *at least as secure as* another cryptosystem $\mathcal{C}'$ if for all attackers $\mathcal{A}$ on $\mathcal{C}$ there is an attacker $\mathcal{A}'$ on $\mathcal{C}'$ achieving the same goal as $\mathcal{A}$. This definition exactly captures the idea that, if there is no attack on $\mathcal{C}$ being more successful than any attack on $\mathcal{C}'$, then, clearly, $\mathcal{C}$ provides the same security as $\mathcal{C}'$. For a precise definition, it needs, however, to be specified what it means that an attacker $\mathcal{A}$ achieves the same goal as another attacker $\mathcal{A}'$.

Note that, in a realistic setting, the entities interacting with the cryptosystem (the honest parties as well as the adversary) are generally not isolated. They are rather embedded in an *environment* $\tilde{\mathcal{E}}$ which is a system providing interfaces to each of them. In order to make statements about the occurrence of certain events[17], it is additionally assumed that $\tilde{\mathcal{E}}$ generates some (final) binary output. For convenience, we will regard the environment $\tilde{\mathcal{E}}$ together with the honest parties as an extended environment system $\mathcal{E}$ (cf. Fig. 3). Let $\mathcal{C}$ and $\mathcal{C}'$ be two cryptosystems, and let Σ be an access structure.

Definition 1. *$\mathcal{C}$ is said to be* at least as secure as *$\mathcal{C}'$ with respect to Σ, denoted $\mathcal{C} \overset{\Sigma}{\succcurlyeq} \mathcal{C}'$, if for all pairs (P, A) in the access structure Σ and 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*

[14] The same term is often used in the context of secret sharing, where it has a different meaning.

[15] Note that it is not required that $P \cup A = \{1, \ldots, m\}$, i.e., there might be "unused" interfaces which are not connected to a honest party nor to the adversary. In this case, one might think of a predefined dummy system being attached to such interfaces.

[16] As an example, let $\mathcal{C}$ be a cryptosystem for n parties which is secure as long as not more than t of these parties are corrupted. Describing $\mathcal{C}$ as a system with n interfaces, one for each party, this means that certain conditions hold with respect to the access structure containing all pairs $(P, \bar{P})$ where P is a subset of $\{1, \ldots, n\}$ having at least $n - t$ elements, and where $\bar{P}$ is its complement.

[17] E.g., an event indicating that an attacker has achieved a certain goal.

the probability distributions of the binary outputs of $\mathcal{E}(\mathcal{C},\mathcal{A})$ *and* $\mathcal{E}(\mathcal{C}',\mathcal{A}')$*,*

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

is negligible in the security parameter k*.*

Similarly, $\mathcal{C}$ *is* computationally at least as secure as $\mathcal{C}'$, *denoted* $\mathcal{C} \overset{\Sigma}{\succsim}_{c} \mathcal{C}'$*, if, additionally,* $\mathcal{E}$*,* $\mathcal{A}$*, and* $\mathcal{A}'$ *are efficient algorithms.*[18]

We will be concerned with general relations between systems, e.g., whether a resource $\mathcal{S}$ can be replaced by another resource $\mathcal{T}$ (without affecting the security of a cryptosystem based on these resources). These relations not only depend on the input/output behavior of the considered resources, but also on the way how the honest parties and a possible adversary might access the resource.[19]

This motivates the following definition: An interface of a system $\mathcal{S}$ is said to be *private with respect to an access structure* Σ if, according to Σ, the interface is only used by honest parties, while an adversary can never have access to it.[20] Similarly, an interface is called *public with respect to* Σ if the interface is not used by the honest parties, but the adversary has access to it.[21]

In the following, speaking of a *resource* $\mathcal{S}$, we mean a system together with a specification, defining for each of its interfaces whether it is private or public. If the public and private interfaces of the resource $\mathcal{S}$ correspond to the public and private interfaces with respect to a given access structure Σ, then Σ is said to be *compatible* with $\mathcal{S}$.

In this paper, we will consider resources whose interfaces are all either private or public. For convenience, multiple interfaces of the same type (public or private) will be regarded as one single interface, i.e., we will restrict ourselves to resources with only *one* private and *one* public interface.

Note that, by definition, the private interface (usually denoted $\mathcal{V}^1$) of a resource $\mathcal{V}$ is only accessed by honest parties, while the public interface ($\mathcal{V}^2$) is not used by these parties. Any construction $\mathcal{B}$ of the honest parties based on $\mathcal{V}$ (e.g., a cryptosystem using $\mathcal{V}$ as a component) thus results in a new resource $\mathcal{W}$, denoted as $\mathcal{B}(\mathcal{V})$, whose private interface $\mathcal{W}^1$ is specified by $\mathcal{B}(\mathcal{V}^1)$ while the public interface $\mathcal{W}^2$ of $\mathcal{W}$ is identical to the public interface $\mathcal{V}^2$ of the original resource $\mathcal{V}$ (cf. Fig. 4).

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.

[18] In a computational setting, it is essential that the output generated by the environment is binary. Otherwise, two outputs might be different while not being distinguishable by any efficient algorithm.

[19] To see this, recall the example mentioned at the beginning of this subsection, where the construction of a private channel based on a broadcast channel for n parties is considered. As long a an opponent has access to the original broadcast channel, it is obviously impossible to obtain privacy. The given reduction thus only works if all pairs in the access structure are of the form $(P,\emptyset)$ where $\emptyset$ is the empty set, i.e., if there is no adversary.

[20] Formally, for a private interface with index k: $\forall(P,A)\in\Sigma : k\notin A \quad \wedge \quad \exists(P,A)\in\Sigma : k\in P$.

[21] Formally, for a public interface with index k: $\forall(P,A)\in\Sigma : k\notin P \quad \wedge \quad \exists(P,A)\in\Sigma : k\in A$.

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 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 only have a private interface.

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 interfaces of the resources are private, i.e., a possible opponent has no direct access to any additional information being correlated with the behavior of the systems.

4.2 Generalization to Indifferentiability

We will now extend the definition of indistinguishability to resources which additionally have a public interface (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 interface 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 again $\mathcal{S} = (\mathcal{S}_k)_{k\in\mathbb{N}}$ and $\mathcal{T} = (\mathcal{T}_k)_{k\in\mathbb{N}}$ be two resources both of them having a private and a public interface with index 1 and 2, respectively (cf. Fig. 5).

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*

$$\left|\mathrm{Prob}[\mathcal{D}(\mathcal{S}_k^1, \mathcal{S}_k^2) = 1] - \mathrm{Prob}[\mathcal{D}(\mathcal{T}_k^1, \mathcal{P}(\mathcal{T}_k^2)) = 1]\right|$$

is negligible in k. [22] *The indifferentiability is* computational*, denoted $\mathcal{S} \sqsubset_c \mathcal{T}$, if for $\mathcal{D}$ and $\mathcal{P}$ only computationally efficient algorithms are considered.*

[22] (In)differentiability can equivalently be described in terms of (a special type of) a proof system: Consider a *verifier* (which takes the role of the distinguisher $\mathcal{D}$) interacting with the private interface of a blackbox $\mathcal{B}$ (where either $\mathcal{B} = \mathcal{S}$ or $\mathcal{B} = \mathcal{T}$) and a *prover* having access to the public interface of $\mathcal{B}$. Then, $\mathcal{S}$ is *differentiable* from $\mathcal{T}$ if, given that $\mathcal{B} = \mathcal{S}$, there is a prover being able to *convince* the verifier of the fact that $\mathcal{B} = \mathcal{S}$.

Note that indistinguishability is a special (symmetric) case of indifferentiability. Indeed, if the resources have no public interface, 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. One important difference is, however, the asymmetric nature of indifferentiability. 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 having one private and one public interface.

Theorem 1. *Let $\mathcal{C}$ range over the set of all cryptosystems. Then, for any access structure Σ being compatible with $\mathcal{S}$ and $\mathcal{T}$,*

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

The same equivalence holds when "$\sqsubset$" and "$\overset{\Sigma}{\succ}$" are replaced by "$\sqsubset_c$" and "$\overset{\Sigma}{\succ}_c$", 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 $\mathcal{C}(\mathcal{S})$.

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.

Proof. To simplify the notation, set

$$d_{\mathcal{D},\mathcal{P}}(k) := |\mathrm{Prob}[\mathcal{D}(\mathcal{S}_k^1, \mathcal{S}_k^2) = 1] - \mathrm{Prob}[\mathcal{D}(\mathcal{T}_k^1, \mathcal{P}(\mathcal{T}_k^2)) = 1]|$$

where $\mathcal{D}$ is a distinguisher and $\mathcal{P}$ a system interacting with the systems $\mathcal{S}$ and $\mathcal{T}$ (having a private and a public interface with index 1 and 2, respectively) as specified by Fig. 5. Similarly, define

$$e_{\mathcal{E},\mathcal{C},(P,A),\mathcal{A},\mathcal{A}'}(k) := |\mathrm{Prob}[\mathcal{E}(\mathcal{C}(\mathcal{S}_k^1), \mathcal{A}(\mathcal{S}_k^2)) = 1] - \mathrm{Prob}[\mathcal{E}(\mathcal{C}(\mathcal{T}_k^1), \mathcal{A}'(\mathcal{T}_k^2)) = 1]|$$

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, according to $(P, A) \in \Sigma$ (see Fig. 2 and Fig. 3).

The statement of the theorem can then be rewritten as

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

Let us start with the first implication ("$\Longrightarrow$"), which is proven by contradiction: Assume that there exists a cryptosystem $\mathcal{C}$, a pair $(P, A) \in \Sigma$ (defining the access of the honest parties and the adversary to $\mathcal{C}$), an environment $\mathcal{E}$, and an attacker $\mathcal{A}$ such that for all attackers $\mathcal{A}'$ the difference $e_{\mathcal{E},\mathcal{C},(P,A),\mathcal{A},\mathcal{A}'}(k)$ is non-negligible in k. Let the distinguisher $\mathcal{D}$ be defined as the system resulting from $\mathcal{C}$, $\mathcal{E}$, and $\mathcal{A}$ being combined according to (P, A) (cf. Fig. 6(a)). Furthermore, for any system $\mathcal{P}$, let the attacker $\mathcal{A}'$ be defined as $\mathcal{A}(\mathcal{P})$ (cf. Fig. 6(b)). The two settings involving the system $\mathcal{S}$ (represented in Fig. 6(a) by solid lines and dashed lines, respectively) as well as the two settings involving the system $\mathcal{T}$ (Fig. 6(b)) are then obviously equivalent, i.e., the probabilities of their outputs are equal. We thus have $d_{\mathcal{D},\mathcal{P}}(k) = e_{\mathcal{E},\mathcal{C},(P,A),\mathcal{A},\mathcal{A}'}(k)$, i.e., $d_{\mathcal{D},\mathcal{P}}(k)$ is non-negligible.

The second implication ("$\Longleftarrow$") is as well proven by contradiction: Assume that there is a distinguisher $\mathcal{D}$ whose advantage $d_{\mathcal{D},\mathcal{P}}(k)$ for all systems $\mathcal{P}$ is non-negligible in k. Let the cryptosystem $\mathcal{C}$ be identical to $\mathcal{D}$,[23] and define both the environment $\mathcal{E}$ and the attacker $\mathcal{A}$ as a trivial system simply forwarding all queries (Fig. 7(a)). Then, for any attacker $\mathcal{A}'$, set $\mathcal{P} := \mathcal{A}'$ (Fig. 7(b)). Again, the two settings involving the system $\mathcal{S}$ (Fig. 7(a)) as well as the two settings involving the system $\mathcal{T}$ (Fig. 7(b)) are equivalent, i.e., $e_{\mathcal{E},\mathcal{C},(P,A),\mathcal{A},\mathcal{A}'}(k)$ equals $d_{\mathcal{D},\mathcal{P}}(k)$ and is thus non-negligible. □

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 each having a private ($\mathcal{U}^1$ and $\mathcal{V}^1$) and a public ($\mathcal{U}^2$ and $\mathcal{V}^2$) interface.

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 deterministic[24] (computationally efficient) algorithm $\mathcal{B} \in \Gamma(\mathcal{V}^1/\mathcal{U}^1)$, making calls to the private interface of $\mathcal{V}$, such that the resulting resource $\mathcal{W}$ (cf. Fig. 4), having private interface $\mathcal{W}^1 = \mathcal{B}(\mathcal{V}^1)$ and public interface $\mathcal{W}^2 = \mathcal{V}^2$, satisfies $\mathcal{W} \sqsubset \mathcal{U}$ ($\mathcal{W} \underset{c}{\sqsubset} \mathcal{U}$).*

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, for any access structure Σ being compatible with $\mathcal{U}$ and $\mathcal{V}$,*

$$\mathcal{U} \to \mathcal{V} \quad \Longleftrightarrow \quad \exists\, \mathcal{B} \in \Gamma(\mathcal{V}^1/\mathcal{U}^1) : \forall\, \mathcal{C} : \mathcal{C}(\mathcal{B}(\mathcal{V})) \overset{\Sigma}{\succ} \mathcal{C}(\mathcal{U}).$$

The same statement holds when "$\to$" and "$\overset{\Sigma}{\succ}$" are replaced by "$\underset{c}{\to}$" and "$\overset{\Sigma}{\underset{c}{\succ}}$", respectively.

6 A Sufficient Criterion for Irreducibility

The following theorem gives an easily verifiable sufficient criterion for a resource $\mathcal{U}$ not to be reducible to another 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.

We will consider resources with one private and one public interface which are both specified by the *same* random function[25]. Let $\mathcal{U} = (\mathcal{U}_k)_{k\in\mathbb{N}}$ and $\mathcal{V} = (\mathcal{V}_k)_{k\in\mathbb{N}}$ be such resources, and assume that the costs for accessing their private interface are given by c and c', respectively, where, 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, $\mathcal{U}$ is not reducible to $\mathcal{V}$ if $h^{\infty}_{\mathcal{U}_k}(t)$ grows "sufficiently faster than" $h^{0}_{\mathcal{V}_k}(t)$.

[23] Motivated by a construction given in [5], 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.

[24] This is no restriction of generality, but implies that all (private and public) randomness to be used by $\mathcal{B}$ has to be defined explicitly as a part of the system $\mathcal{V}$.

[25] This implies that on the same inputs, the outputs are identical at both interfaces.

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 $\mathcal{U} \not\twoheadrightarrow \mathcal{V}$. If, additionally, $h^{\infty}_{\mathcal{U}_k}(t)$ grows at least linearly in t, and $h^0_{\mathcal{V}_k}(t)$ grows at most polynomially in t and k, then $\mathcal{U} \not\hookrightarrow \mathcal{V}$.*

The proof is a direct generalization of the proof of the separation result presented in Section 2.

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

$$\bar{c}_k(x) \leq p(k, c_k(x)) \ . \tag{1}$$

Similarly to the proof presented in Section 2, we will 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. 5 where $\mathcal{S} = \mathcal{B}(\mathcal{V})$ and $\mathcal{T} = \mathcal{U}$), called $\mathcal{D}^1$ and $\mathcal{D}^2$, respectively .

Let, for $r \in \mathbb{N}$, 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 $\mathcal{D}^2$ and stores the received answers $Y'_1, \ldots, Y'_l$ (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 $\bar{Y}_i$. Thereby, any query $X' \in \{1, \ldots, l\}$ of (the simulated) $\mathcal{B}$ to $\mathcal{V}$ is answered by the corresponding stored value $Y'_{X'}$. If $X' > l$, $\mathcal{D}$ stops with output 0. The same test inputs X_i are then sent to interface $\mathcal{D}^1$, 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 $\mathcal{D}$ has however to be slightly modified in order to avoid the following technical problem: The stored values $Y'_1, \ldots, Y'_l$ 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 $Y'_1, \ldots, Y'_l$ as answers for $\mathcal{B}$'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.

We have to show the following properties of $\mathcal{D}$:

(a) $\mathcal{D}(\mathcal{U}^1, \mathcal{P}(\mathcal{U}^2))$ outputs 1 with probability being negligible in k.
(b) $\mathcal{D}(\mathcal{B}(\mathcal{V}^1), \mathcal{V}^2)$ outputs 1 with certainty.
(c) In the computational case, $\mathcal{D}(\cdot,\cdot)$ is efficient.

$\mathcal{D}$ 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 $Y'_1, \ldots, Y'_l$ used for the simulation of $\mathcal{B}(\mathcal{V})$ (and since, by definition, $\mathcal{B}$ 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 $p_k(\cdot) := p(k, \cdot)$, $p(\cdot, \cdot)$ being 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 construction, $\mathcal{D}$ performs tests on inputs $X_i := i$ for $i = 1, \ldots, n$ where $n := (\bar{h}_k)^{-1}(l+k)$. Hence,

$$c_k(x) \leq (h^\infty_{\mathcal{U}_k})^{-1}(l+k)$$

holds for all $x = 1, \ldots, n$. By assumption, the costs c and $\bar{c}$ (of $\mathcal{U}$ and $\mathcal{B}(\mathcal{V})$, respectively) satisfy condition (1). The costs $c'_k(x')$ of $\mathcal{V}$ for each potential query x' of $\mathcal{B}$ to $\mathcal{V}$ are thus bounded by

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

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 $Y'_1, \ldots, Y'_l$ contains the answers to all queries x' of $\mathcal{B}$ to $\mathcal{V}^1$. Clearly, this is the case if $l \geq l'$, which is true if l satisfies inequality (2).

It remains to be proven 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 less 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).

$\mathcal{D}$ satisfies property (c). It has to be shown that in the computational case (where, by assumption, $h^0_{\mathcal{V}_k}(t)$ only grows polynomially in t and k), efficiency of $\mathcal{B}$ implies efficiency of $\mathcal{D}$. By construction (since the simulation of $\mathcal{B}$ is only run for a certain number of steps), if $\mathcal{B}$ is efficient when interacting with $\mathcal{V}^1$, then each simulation of $\mathcal{B}$ by $\mathcal{D}$ is efficient as well (for any arbitrary system $\mathcal{P}$ connected to $\mathcal{D}^2$). Furthermore, the number n of simulations is, by the assumption on $h^\infty_{\mathcal{U}_k}$, itself bounded linearly in l. It thus remains to be shown that the minimal l satisfying inequality (2) only grows polynomially in k. The straightforward calculation is omitted. □

7 Applications

7.1 Random Oracles, Asynchronous Beacons, and Finite Random Strings

We will now apply the framework presented in the previous sections to the examples mentioned in the introduction (Section 1), i.e., to random oracles, beacons and finite random strings. These are modeled as a 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 $\mathcal{X} = \{0,1\}^*$ and output set $\mathcal{Y} = \{0,1\}$).[26] 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, $c(x) := |x|$. (In the following, we only need an upper bound for the costs of a random oracle, i.e., $c(x) \leq |x|$.) For a *asynchronous beacon* $\mathcal{Q}$, R is also an infinite bitstring, but the costs for the queries are higher, namely $c(x) := x$, where x is interpreted as a natural number. On the other hand, for a *finite random string* $\mathcal{F}$, the length 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 (which possibly depends on the security parameter k).[27] 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). However, it follows directly from Theorem 3 that the inverse reductions are not possible.

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

$$\overline{\mathcal{R}} \not\to \mathcal{Q} \qquad \text{and} \qquad \overline{\mathcal{Q}} \not\to \mathcal{F}.$$

The proof mainly consists of the computation of the entropies of the considered objects which then allows to apply Theorem 3.

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

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

and similarly, for an asynchronous beacon,

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

(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

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

[26] 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 such random function 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).

[27] Equivalently, one could take any polynomial in $|x|$ and $|k|$.

(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.

With these expressions for the entropies, the assertion follows directly from Theorem 3. $\square$

Together with Theorem 2, it follows from this result 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 [5] stating that random oracles can not be implemented. In view of the highly involved arguments of [5] 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 an adversary can obtain side-information about secret parameters.

References

1. 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.
2. 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 *Lecutre Notes in Computer Science*, pages 399–416. Springer-Verlag, 1996.
3. R. Canetti. Security and composition of multi-party cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.
4. 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.
5. 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.
6. 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.
7. 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.
8. 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.
9. U. Maurer. Indistinguishability of random systems. In Lars Knudsen, editor, *Advances in Cryptology — EUROCRYPT '02*, volume 2332 of *Lecture Notes in Computer Science*, pages 110–132. Springer-Verlag, 2002.
10. S. Micali. CS proofs. In *Proc. 35th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 436–453. IEEE, 1994.
11. T. Okamoto. Provably secure and practical identification scheme and corresponding signature scheme. In *Advances in Cryptology — CRYPTO'92*, volume 740, pages 31–53. Springer-Verlag, 1992.

12. 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.
13. B. Pfitzmann and M. Waidner. A model for asynchronous reactive systems and its application to secure message transmission. Technical Report 93350, IBM Research Division, Zürich, 2000.
14. D. Pointcheval and J. Stern. Security proofs for signature schemes. In *Advances in Cryptology — EUROCRYPT'96*, volume 1070 of *Lecutre Notes in Computer Science*, pages 387–398. Springer-Verlag, 1996.
15. C. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

Appendix

Figures

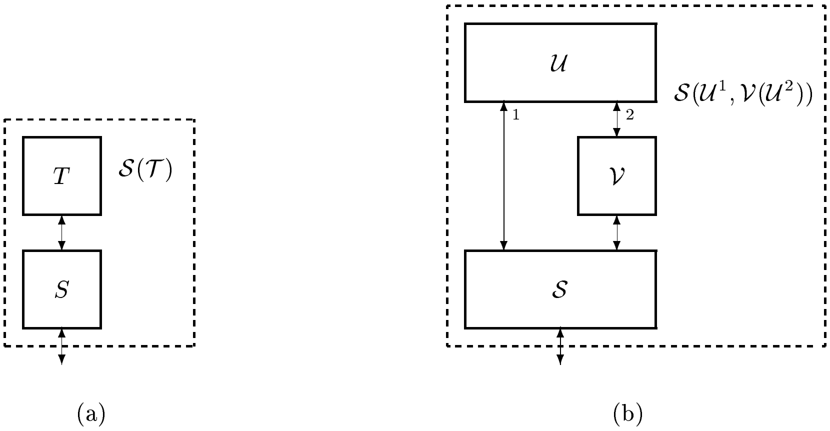

Fig. 1. Composition of systems.

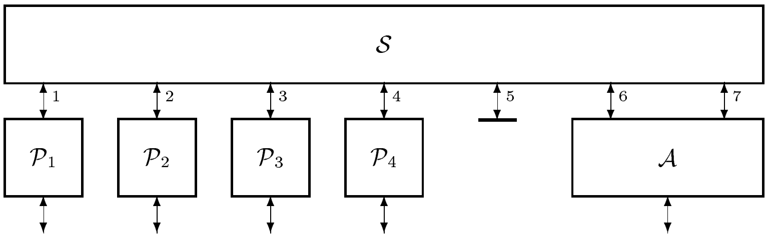

Fig. 2. Access of parties $\mathcal{P}_1, \ldots, \mathcal{P}_4$ and adversary $\mathcal{A}$ to system $\mathcal{S}$, specified by $(P, A) = (\{1, 2, 3, 4\}, \{6, 7\})$.

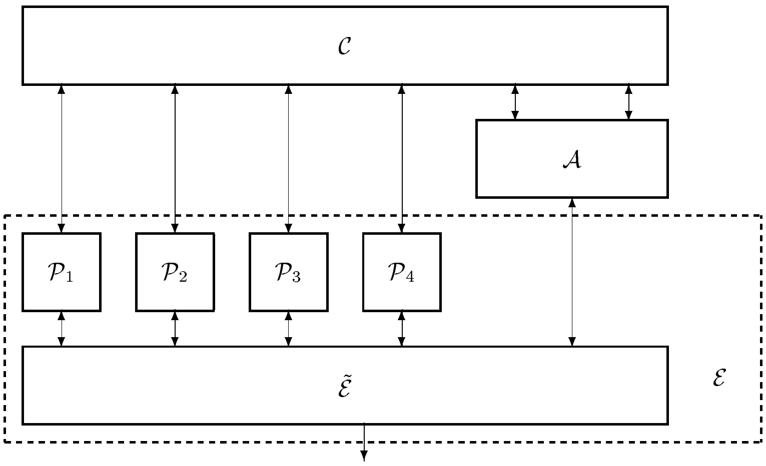

Fig. 3. Cryptosystem $\mathcal{C}$ with parties $\mathcal{P}_1, \ldots, \mathcal{P}_4$ and adversary $\mathcal{A}$ embedded in environment $\tilde{\mathcal{E}}$.

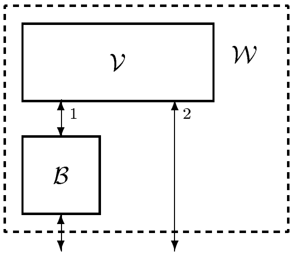

Fig. 4. Construction of new resource $\mathcal{W} = \mathcal{B}(\mathcal{V})$ from original resource $\mathcal{V}$.

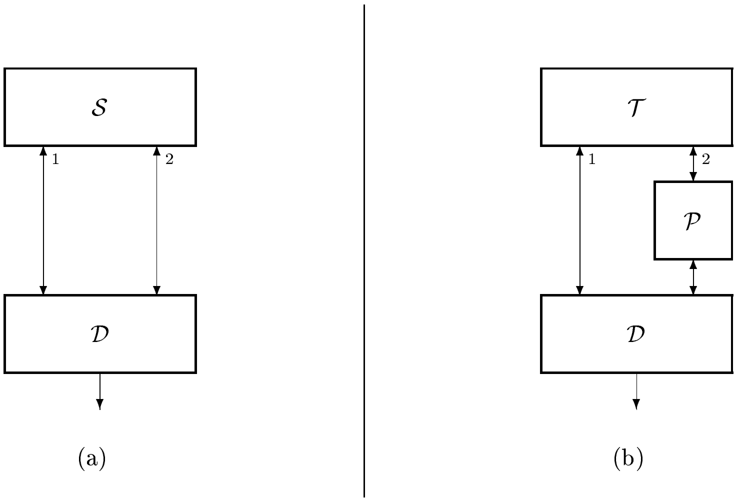

Fig. 5. Indifferentiability: The distinguisher $\mathcal{D}$ for differentiating $\mathcal{S}$ from $\mathcal{T}$ is either connected to the system $\mathcal{S}$ or the system $\mathcal{T}$. In the first case (a), $\mathcal{D}$ has direct access to the private (index 1) and the public (index 2) interface of $\mathcal{S}$, while, in the latter case (b), the access to the public interface of $\mathcal{T}$ is replaced by an arbitrary intermediate system $\mathcal{P}$.

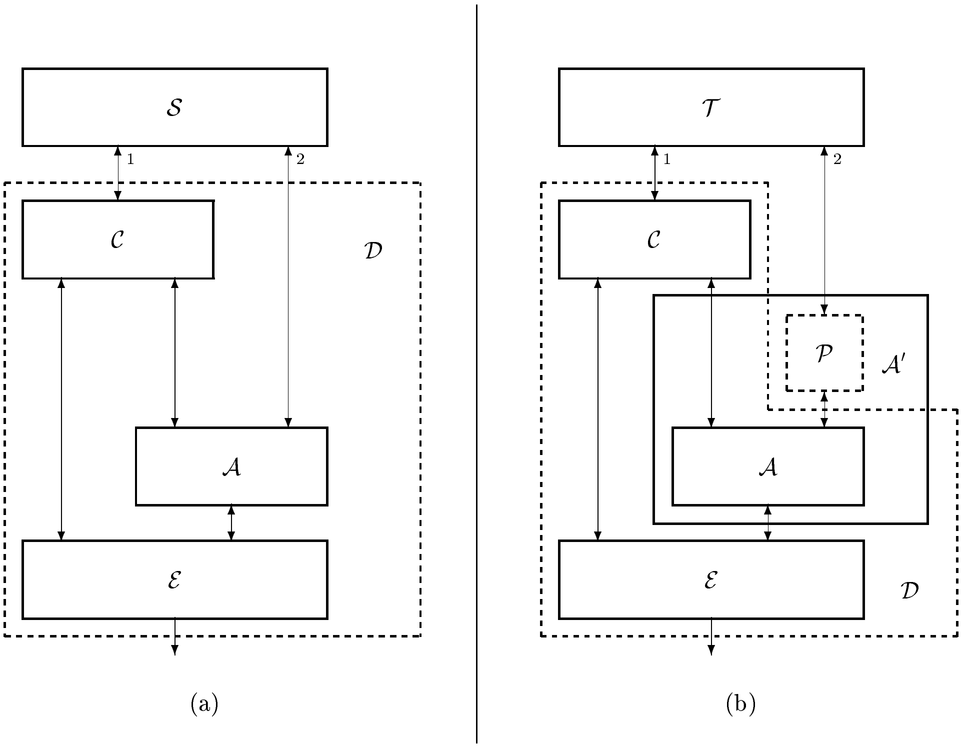

Fig. 6. Illustration for proof of Theorem 1 ("$\Longrightarrow$").

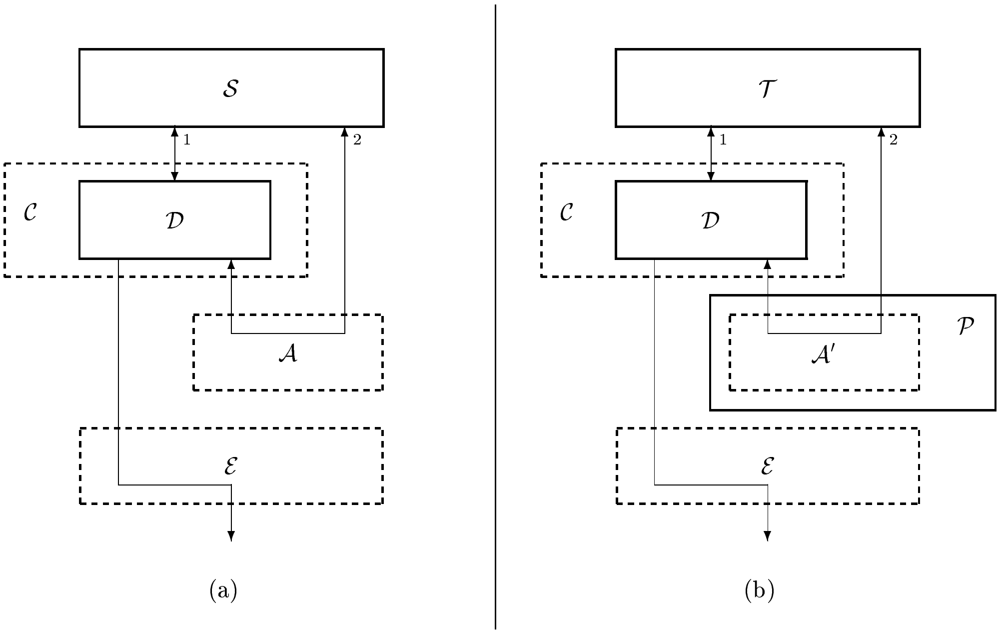

Fig. 7. Illustration for proof of Theorem 1 ("$\Longleftarrow$").