On the Relation Between the Ideal Cipher and the Random Oracle Models

Yevgeniy Dodis* and Prashant Puniya

Courant Institute of Mathematical Sciences,
New York University
{dodis,puniya}@cs.nyu.edu

Abstract. The Random Oracle Model and the Ideal Cipher Model are two of the most popular idealized models in cryptography. It is a fundamentally important practical and theoretical problem to compare the relative strengths of these models and to see how they relate to each other. Recently, Coron et al. [8] proved that one can securely instantiate a random oracle in the ideal cipher model. In this paper, we investigate if it is possible to instantiate an ideal block cipher in the random oracle model, which is a considerably more challenging question. We conjecture that the *Luby-Rackoff construction* [19] with a sufficient number of rounds should suffice to show this implication. This does not follow from the famous Luby-Rackoff result [19] showing that 4 rounds are enough to turn a pseudorandom function into a pseudorandom permutation, since the results of the intermediate rounds are known to everybody. As a partial step toward resolving this conjecture, we show that random oracles imply ideal ciphers in the *honest-but-curious model*, where all the participants are assumed to follow the protocol, but keep all their intermediate results. Namely, we show that the *Luby-Rackoff construction* with a superlogarithmic number of rounds can be used to instantiate the ideal block cipher in any honest-but-curious cryptosystem, and result in a similar honest-but-curious cryptosystem in the random oracle model. We also show that securely instantiating the ideal cipher using the Luby Rackoff construction with upto a logarithmic number of rounds is equivalent in the honest-but-curious and malicious models.

1 Introduction

Designing provably secure as well as efficient cryptographic protocols is never an easy task. When one tries to achieve provable security without making any assumptions, it often comes at the expense of simplicity and efficiency of the design. On the other hand, practical and efficient schemes are often based on heuristics that cannot be justified with a formal proof. In the late 1990s, this problem was addressed and several ideas were proposed to strike a balance between these two conflicting requirements.

Random Oracle Model. One of these was the formalization of the well known *Random Oracle Model* (ROM) by Bellare and Rogaway [3]. In this model, we

* Supported in part by NSF career award CCR-0133806 and NSF grant CCR-0311095.

assume the existence of a publicly accessible ideal random function and prove protocol security based on this assumption. As was shown by a huge body of literature (for a small set of representative examples, see [3, 6, 4, 15, 24, 25]), the ROM often allows one to design very simple, intuitive and efficient protocols for many tasks, while simultaneously providing a seemingly convincing security guarantee for such practical constructions. Of course, in practice an ideal random function is instantiated by a concrete, "heuristically-secure" hash function, such as one of the *SHA* functions. The hope of the security of such a substitution comes from the optimistic belief that, — although no security proof is currently found with the heuristic hash function, — the only way such composition can fail is if some some unexpected inter-dependency between a protocol and the *code* of a concrete hash function is found. For practical protocols and real-life, "messy" hash functions, it seems unlikely that such unexpected inter-dependency should be found, at least not without directly attacking a carefully-designed heuristic hash function, which is also considered unlikely. On the other side, in theory such security proofs in the ROM have came under scrutiny, after a series of results showed artificial schemes that are provably secure in the ROM, but are uninstantiable in the standard model [10, 22, 16, 11, 2]. Still, none of these results directly attack any of the widely used cryptographic schemes, such as OAEP [6] or PSS [4], that rely on secure hash functions. In particular, all the practical applications of the random oracle methodology still appear to be "plausibly secure". Additionally, in some cases the protocols in the ROM came before and *influenced* the first (often slower) solutions in the standard model, and in some other cases the ROM solutions are still the *only* known solutions. To summarize, the random oracle model remains a useful and popular tool in the protocol design.

Ideal Cipher Model. Another example of such an ideal assumption model is the *Ideal Cipher Model* (ICM) (also known as the "Shannon Model"). In this model, we assume the existence of a publicly accessible Ideal Block Cipher. This is a block cipher, with a k bit key and a n bit input, that is chosen uniformly from all block ciphers of this form. All parties in the ICM can make both forward (encryption) or inverse (decryption) queries to the ideal block cipher. One proves the security of a cryptosystem under this assumption, and then instantiates the ideal block cipher with a practical block cipher construction, such as AES. Although the ICM is not as popular as the random oracle model, there are still several examples of schemes where this model has been used [5, 13, 14, 17, 18].

Several questions have been raised regarding security in the ideal cipher model. Existing block cipher constructions, such as DES, AES etc. are vulnerable to related key attacks and have distinguishing patterns that are unlikely to occur in a random permutation. Hence it may not be entirely secure to use these constructions to instantiate the ideal block cipher. As in the case of the random oracle model, uninstantiable schemes that are secure in the ideal cipher model have also been presented (see [1]). But, all these problems withstanding, the ideal cipher model does provide security against generic attacks that do not exploit weaknesses of the underlying block cipher.

Comparing The Models. From a theoretical viewpoint, it is interesting to compare different ideal assumption models (such as ROM and ICM). That is, compare two ideal assumption models to see which one provides a better security guarantee. There was no satisfactory definition that captured this idea until recently. In TCC 2004, Maurer et al [20] proposed an extension of the classical notion of indistinguishability, called *indifferentiability*. Based on this notion of indifferentiability, Coron et al [8] gave the definition of an "indifferentiable construction" of one ideal primitive (F) using another (G). If a construction satisfies this definition, then any application that is provably secure in the former ideal model (F) remains provably secure in the latter model (G) as well, when instantiated using this construction.

It is an interesting question to analyze the relationship between the *Random Oracle Model* and the *Ideal Cipher Model* using this notion of indifferentiability. It had been believed for quite some time that it should be possible to instantiate a *random oracle* in the *ideal cipher model*. This is because an ideal block cipher seems to be a much stronger primitive than a random oracle, as it seems plausible that one can construct "unstructured" functions from permutations. this In [8], a formal proof of this conjecture was given. The authors analyzed the Merkle-Damgård construction [12, 21] for extending the domain of a random function in the indifferentiability scenario. The Merkle-Damgård construction is the basis of almost all practical hash functions, such as SHA or MD5. It was shown in [8] that, although the plain Merkle-Damgård construction does *not* work in extending the domain of a random oracle in the indifferentiability model, several slight (and easily implementable) modifications of this construction formally satisfy the indifferentiability requirement. In fact, they also extended these constructions to the *ideal cipher model* and showed that, by using the *Davies Meyer hash function* [26] in place of a "fixed-size" random oracle, any of these modified constructions still satisfy the indifferentiability definition. This result, in turn, implies that *a random oracle can be securely instantiated in the ideal cipher model*.

What about the other direction of this question? *Can one securely instantiate an ideal cipher in the random oracle model?* This direction seems much more difficult to tackle. Actually, it is widely believed that a positive answer holds in this direction too [9]. In fact, it is conjectured that, with a sufficient number of rounds, the *Luby-Rackoff (LR) construction* [19] (with independent random oracles, indexed by the ideal cipher key and the round number, as round functions) is a secure construction of an ideal block cipher in the random oracle model.[1] In spite of this, there has not been much progress in getting a formal proof of this conjecture.

Our Main Result. In this paper, we take a step toward finding such a proof. Namely, we will show that the Luby-Rackoff construction works in the *honest-but-curious* model, where all the participants are assumed to follow the pre-

[1] We notice that the famous Luby-Rackoff result [19], showing that 4 rounds are enough to turn a pseudorandom function into a pseudorandom permutation, is not applicable here, since it crucially relies on the secrecy of the intermediate round values, while in our setting such intermediate round values are public.

scribed protocols, but keep all the intermediate results (such as the intermediate round values in the LR construction). Namely, we show that the LR-construction (with a superlogarithmic number of rounds in the security parameter) can be used to instantiate the ideal block cipher in *any* honest-but-curious cryptosystem, and result in a similar honest-but-curious cryptosystem in the random oracle model. While weaker that result in the malicious model, we stress that the conclusion works for *any application in the honest-but-curious model*, even a "maliciously chosen one". In essence, we are using the honest-but-curious aspect only in assuming that the participants will not use the random oracle for purposes other than honestly evaluating the LR construction on *adversarially chosen* points. We now describe our results in more detail.

1.1 Our Results in More Detail

We will start by recalling the definition of *indifferentiability* of a construction of an ideal primitive. This is the same definition as described in [8]. We will then describe what it means to implement an ideal primitive G using an ideal primitive F in the "honest-but-curious model". We will present a restricted version of the definition of general indifferentiability that captures this notion, which we will call *indifferentiability in the honest-but-curious model*. This definition is weaker than general indifferentiability, but is considerably stronger than the classical notion of indistinguishability (see below). We will also describe special types of constructions, which we call *transparent constructions*, for which this restricted definition is equivalent to general indifferentiability.

Once we have a suitable definition, we will describe the *random permutation model* where we assume the existence of a publicly accessible random permutation π (and its inverse π^{-1}). Note that this can be thought of as a very special case of the *ideal block cipher*, where the key space has a single element. We will show that if we can find an indifferentiable construction of a *random permutation* from a random oracle, it can be easily extended to get an indifferentiable construction of an *ideal block cipher* from a random oracle. This is simply done by prepending the key to the block cipher to the input of the random oracle. Thus, it is (necessary and) sufficient to study constructions of a single random permutation from a random oracle.

We will then describe a construction of a random permutation from a random oracle: namely, the *LR-construction* described above, where we derive the round functions from the *random oracle* (indexed by the round number). We conjecture that the LR-construction is indifferentiable from a *random permutation*, with a sufficient number of rounds. As we said, though, we will not be able to prove this result in general. Our main result, however, will prove this implication in the honest-but-curious model, *as long as the number of rounds is super-logarithmic in the security parameter* λ. The proof of this theorem is quite non-trivial, and will essentially show that any distinguisher needs to make an exponential number of queries to have a non-negligible chance of telling apart this construction from a true random permutation in the honest-but-curious indifferentiability scenario.

We conjecture that our result is sub-optimal in a sense that the LR construction seems to be secure even with a "large enough" *constant* number of rounds (see below on what large enough could be), and even in the malicious model. However, we show its "optimality" in the following sense: we prove that for upto a logarithmic number of rounds the LR-construction is a *transparent construction*. Thus, short of resolving our conjecture in the malicious model, any improvement in the number of rounds even in the *honest-but-curious* model will right away imply the same result in the *malicious* model as well. From a positive spin, for upto logarithmic number of rounds one can *without loss of generality* concentrate on the honest-but-curious model (although we have no indication if such proof will be any simpler). From a negative side, we show that for super-logarithmic number of rounds the LR-construction is *provably not transparent*, which means that our positive result in the honest-but-curious model does not trivially imply the same result in the malicious model.

Finally, we mention that for any less than 6 rounds, the LR-construction is not an indifferentiable construction of a random permutation. (The same will also hold in the honest-but-curious model since for less than 6 rounds the LR-construction is a transparent construction.) Aside from showing that at least 6 rounds are needed, this result can be seen as a separation between indifferentiability, even in the honest-but-curious model, and the classical notion of indistinguishability. This is because, in [19], Luby and Rackoff proved that for ≥ 4 rounds this construction is *indistinguishable* from a random permutation. To put it differently, even in the context of the LR-construction the ability to observe "intermediate results" gives a noticeable edge to the adversary, partially explaining why the indifferentiability result seems to be much harder to get.

2 Definitions

In this section, we introduce the main notations and definitions that we will use henceforth. An *ideal primitive* is an algorithmic entity that receives a query from one of the parties and responds to the querying party immediately, and which implements some functionality in an ideal fashion. The ideal primitives we will consider in this paper are random oracles and ideal ciphers. A random oracle is an ideal implementation of a function that assigns a uniformly random value (chosen from a prespecified range) to each input. An ideal cipher is an ideal implementation of a block cipher $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$. Each key $k \in \{0,1\}^\kappa$ to the block cipher E defines a random permutation $E_k = E(k, \cdot)$ on $\{0,1\}^n$. The ideal cipher E accepts both forward queries (E) as well as inverse queries (E^{-1}) ($(0, k, m)$ or $(1, k, c)$ resp.).

2.1 Preliminaries

Let us first establish some basic notation that we will be using. We denote the set of all functions $\{0,1\}^n \to \{0,1\}^n$ by F_n and the set of all permutations on $\{0,1\}^n$ by P_n (clearly, $P_n \in F_n$). For a bit string x, $x|_L$ and $x|_R$ denote the left and right halves of x, respectively. $\oplus$ denotes bit by bit XOR of two bit strings.

Definition 1 (Feistel Permutation). *Given a function $f \in F_n$, the Feistel permutation Ψ_f is a permutation in P_{2n} that outputs $x|_R \parallel x|_L \oplus f(x|_R)$ where $x|_L$ and $x|_R$ are the left and right halves of the $2n$ bit input x, respectively.*

It is easy to see that Ψ_f is a permutation in P_{2n} for any function $f \in F_n$. In fact, it is really easy to invert the Feistel permutation as well. Indeed, $\Psi_f^{-1}(S \parallel T) = (f(S) \oplus T) \parallel S$.

Luby and Rackoff [19] define *pseudorandom permutation ensembles* (PPE) to be distributions of permutations that are indistinguishable from the uniform distribution for any efficient distinguisher. When the distinguisher has access to both the forward and inverse permutation, it is called a *strong pseudorandom permutation ensemble* (SPPE). It was proven in [19], that a 3 (4 resp.) round application of the Feistel permutation, with independent round functions in each round is a PPE (SPPE resp.)

2.2 Indifferentiability and the honest-but-curious model

We will use the notion of *indifferentiability* introduced by Maurer et al [20] to define a secure implementation of an ideal primitive. The ideal primitive that we will attempt to implement is an *ideal cipher*. In [8], the notion of indifferentiability was used to define the security of hash functions (as random oracles). Thus the treatment in [8] is suitable for our problem as well. We now briefly recall the main definitions involved here:

Definition 2. *A Turing machine $C_{\mathcal{G}}$ with oracle access to an ideal primitive $\mathcal{F}$ is said to be (t_D, t_S, q, ϵ) indifferentiable from an ideal primitive $\mathcal{G}$ if there exists a simulator S, such that for any distinguisher D it holds that $|Pr[D^{C_{\mathcal{G}},\mathcal{F}} = 1] - Pr[D^{\mathcal{G},S} = 1]| < \epsilon$. The simulator has oracle access to the ideal primitive $\mathcal{G}$ and runs in time t_S. The distinguisher runs in time at most t_D and makes at most q queries.*

It is shown in [20] that if $C_{\mathcal{G}}^{\mathcal{F}}$ is indifferentiable from $\mathcal{G}$, the $C_{\mathcal{G}}^{\mathcal{F}}$ can replace $\mathcal{G}$ in any cryptosystem, and the resulting cryptosystem will be at least as secure in the $\mathcal{F}$ model as in the $\mathcal{G}$ model. See [8] for more details.

The above definition works for any malicious adversary. We will now present a relaxed version of this notion that we will refer to as *indifferentiability in the honest-but-curious model* for reasons that will be clear soon. In the new definition, the distinguisher effectively has active access to only one oracle. To illustrate this, in the $\mathcal{F}$ model the distinguisher can only query the $\mathcal{G}$ construction $C_{\mathcal{G}}^{\mathcal{F}}$, and not the $\mathcal{F}$ oracle. In addition, it also has access to the queries made by the construction $C_{\mathcal{G}}$ to $\mathcal{F}$, which we will denote as the communication transcript $\mathcal{T}_{C_{\mathcal{G}} \leftrightarrow \mathcal{F}}$. Thus the role of the simulator S in the $\mathcal{G}$ model changes from trying to simulate $\mathcal{F}$ in the general indifferentiability (defn. 2), to trying to simulate the communication transcript $\mathcal{T}_{C_{\mathcal{G}} \leftrightarrow \mathcal{F}}$ in $\mathcal{G}$ model. When the distinguisher has access to $C_{\mathcal{G}}$ and $\mathcal{F}$, its queries can be divided into two types. Those for which it does not observe the queries of $C_{\mathcal{G}}$, and those for which it does. In the $\mathcal{G}$ mode, the former queries are sent directly to the $\mathcal{G}$ oracle and the responses of $\mathcal{G}$ are sent back. While the latter queries are made through the simulator S, which forwards the same query to the $\mathcal{G}$ oracle. But apart from sending back $\mathcal{G}$'s response to the distinguisher, it also sends a simulated communication transcript $\mathcal{T}_S$. These two views of the distinguisher are depicted in figure 1.

Definition 3. *A Turing machine $C_\mathcal{G}$ (with oracle access to $\mathcal{F}$) is said to be (t_D, t_S, q, ϵ) indifferentiable from an ideal primitive $\mathcal{G}$ in the honest-but-curious model if there exists a simulator S such that for any distinguisher D it holds that:*

$$\left| Pr\left[D^{C_\mathcal{G}, \mathcal{T}_{C_\mathcal{G} \leftrightarrow \mathcal{F}}} = 1\right] - Pr\left[D^{\mathcal{G}, \mathcal{T}_S} = 1\right]\right| < \epsilon$$

The simulator S simulates the transcript $\mathcal{T}_S$ for queries made by the distinguisher to it and runs in time t_S. The distinguisher D runs in time at most t_D and makes at most q queries to its oracle. The distinguishing advantage ϵ is a negligible function of the security parameter λ. If t_S and q are both polynomial in λ then the construction $C_\mathcal{G}$ is said to be polynomially indifferentiable from $\mathcal{G}$ in the honest-but-curious model.

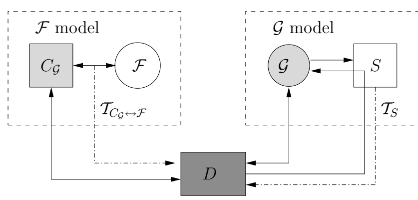

Fig. 1. Indifferentiability in honest-but-curious model: The distinguisher D either interacts with $C_\mathcal{G}$ and gets the transcript $\mathcal{T}_{C_\mathcal{G} \leftrightarrow \mathcal{F}}$ or it interacts with $\mathcal{G}$ and gets the simulated transcript $\mathcal{T}_S$

Note that the simulator S does not make any extra queries to $\mathcal{G}$ apart from forwarding the queries made by the distinguisher D. This fact is crucial since we want the property that the distinguisher should not learn anything from observing the internal functioning of $C_\mathcal{G}$ (i.e. queries made to $\mathcal{F}$), that it cannot learn from an ideal $\mathcal{G}$ itself.

Consider a construction $C_\mathcal{G}$ that is (polynomially) indifferentiable from $\mathcal{G}$ in the honest-but-curious model. Our new definition guarantees that any cryptosystem $\mathcal{P}$, possibly involving honest-but-curious parties, that uses the construction $C_\mathcal{G}$ in the $\mathcal{F}$ model behaves in exactly the same way as it does in the $\mathcal{G}$ model. This fact is formally stated in the following lemma.

Lemma 1. If a construction $C_\mathcal{G}$ using $\mathcal{F}$ is indifferentiable from $\mathcal{G}$ in the honest-but-curious model, as stated in definition 3, then any cryptographic protocol $\mathcal{P}$ (involving honest-but-curious parties possibly) using $C_\mathcal{G}$ in the $\mathcal{F}$ model behaves exactly the same way as in the $\mathcal{G}$ model.

Proof: [also see figure 2] Say there exists a protocol $\mathcal{P} = (\mathcal{P}_{hon}, \mathcal{P}_{cur})$ that behaves differently when using $C_{\mathcal{G}}$ in $\mathcal{F}$ model. $\mathcal{P}_{hon}$ represents the conventional honest parties of the protocol, and $\mathcal{P}_{cur}$ represents the curious ones. We claim that the curious parties $\mathcal{P}_{cur}$ do not gain any extra information when using the construction $C_{\mathcal{G}}$. We will prove this by simulating the view of all parties in $\mathcal{P}$ in the $\mathcal{F}$ model, in the $\mathcal{G}$ model as well. But this is exactly what definition 3 guarantees. We simply replace the construction $C_{\mathcal{G}}$ with $\mathcal{G}$. And we use the simulator S guaranteed by our definition to simulate the transcript $\mathcal{T}_{C_{\mathcal{G}} \leftrightarrow \mathcal{F}}$ for the curious parties $\mathcal{P}_{cur}$. Thus the queries made by the curious parties $\mathcal{P}_{cur}$ are directed through the simulator S, which along with the response of $\mathcal{G}$ adds a fake transcript $\mathcal{T}_S$ for the curious parties. The conventional honest parties $\mathcal{P}_{hon}$ are given direct access to the ideal primitive $\mathcal{G}$. And the indistinguishability of the two scenarios $(C_{\mathcal{G}}, \mathcal{T}_{C_{\mathcal{G}} \leftrightarrow \mathcal{F}})$ and $(\mathcal{G}, \mathcal{T}_S)$ implies that the views of all parties in the protocol remains the same. $\square$

We note here that the notion of "indifferentiability of $C_{\mathcal{G}}$ from $\mathcal{G}$ in the honest

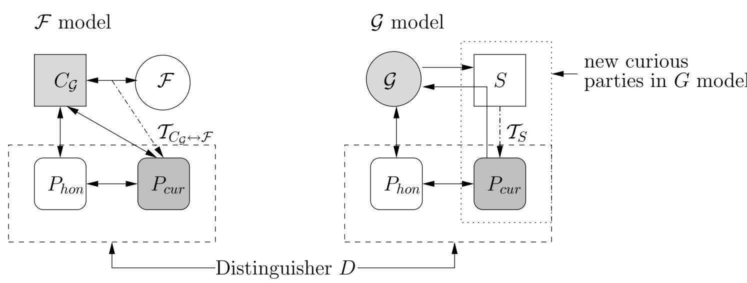

Fig. 2. An idea of the proof of lemma 1. The conventional honest parties P_{hon} along with the curious ones P_{cur} can be seen together as a distinguisher D

but curious model" is at least as strong as (in fact, as we shall see later, strictly stronger than) the notion of "indistinguishability of $C_{\mathcal{G}}$ and $\mathcal{G}$". Clearly, a distinguisher in the indistinguishability scenario will work in the former scenario (def. 3) simply by ignoring the transcripts $\mathcal{T}_{C_{\mathcal{G}} \leftrightarrow \mathcal{F}}$ (or $\mathcal{T}_S$).

2.3 Transparent Constructions

Even though general indifferentiability (definition 2) seems to be much stronger than indifferentiability in the honest-but-curious model (definition 3), we now show that for certain types of constructions these two definitions are, in fact, equivalent.

Definition 4 (Transparent Constructions). *A construction $C_{\mathcal{G}}$ of $\mathcal{G}$ (using oracle access to $\mathcal{F}$) is a (t_E, q_E) transparent construction if there exists a Turing machine E (called an "extracting algorithm") such that for any $x \in dom(\mathcal{F})$ it* *is the case that* $E^{C_{\mathcal{G}}^{\mathcal{F}}, \mathcal{T}_{C_{\mathcal{G}} \leftrightarrow \mathcal{F}}}(x) = \mathcal{F}(x)$. *Here* $\mathcal{T}_{C_{\mathcal{G}} \leftrightarrow \mathcal{F}}$ *denotes the transcript of all the communication between* $C_{\mathcal{G}}$ *and* $\mathcal{F}$. E *runs in time* t_E *and makes at most* q_E *queries to* $C_{\mathcal{G}}^{\mathcal{F}}$ *for any input* x, *while* $dom(\mathcal{F})$ *represents the domain of* $\mathcal{F}$. *And* $|x|$, t_D *and* q_E *are polynomial in the security parameter* λ.

Thus a transparent construction $C_{\mathcal{G}}^{\mathcal{F}}$ is such that it is possible to efficiently compute $\mathcal{F}(x)$ at any input x by making a polynomial number of queries to $C_{\mathcal{G}}$ and observing the communication between $C_{\mathcal{G}}$ and its oracle $\mathcal{F}$.

Lemma 2. If a transparent construction $C_{\mathcal{G}}$ (using $\mathcal{F}$) is (polynomially) indifferentiable from $\mathcal{G}$ in the honest-but-curious model (defn. 3) then it is also (polynomially) indifferentiable from $\mathcal{G}$ (defn. 2).

Proof: Say that a construction $C_{\mathcal{G}}$ is indifferentiable from ideal primitive $\mathcal{G}$ in the honest-but-curious model. Then we have a simulator S_{hon} that successfully fakes the transcript $\mathcal{T}_{C_{\mathcal{G}} \leftrightarrow \mathcal{F}}$ (with $\mathcal{T}_{S_{hon}}$) in the $\mathcal{G}$ model.

First, we will design a simulator S_{mal} for general indifferentiability using the simulator S_{hon}. The simulator S_{mal} needs to simulate the ideal primitive $\mathcal{F}$ in $\mathcal{G}$ model. On getting a query $x \in dom(\mathcal{F})$, S_{mal} uses the extracting algorithm E (for $C_{\mathcal{G}}$) to compute $\mathcal{F}(x)$. The extracting algorithm needs oracle access to the construction $C_{\mathcal{G}}$ and the communication transcript $\mathcal{T}_{C_{\mathcal{G}} \leftrightarrow \mathcal{F}}$. The simulator S_{mal} replaces the construction $C_{\mathcal{G}}$ with the ideal G oracle, which it has access to. And it uses the "honest-but-curious" simulator S_{hon} to produce a fake transcript for E. By definition 3 the extracting algorithm E has no way to tell that it has oracle access to $(\mathcal{G}, \mathcal{T}_{S_{hon}})$ instead of $(C_{\mathcal{G}}, \mathcal{T}_{C_{\mathcal{G}} \leftrightarrow \mathcal{F}})$. This simulator conversion is illustrated in figure 3a.

Now we will show that the simulator S_{mal} designed above actually works. To the contrary, say there is a distinguisher D_{mal} with non-negligible advantage in the general indifferentiability game. Then we will design a distinguisher D_{hon} for the honest-but-curious indifferentiability scenario. D_{hon} simply runs the "malicious" distinguisher D_{mal} and uses the extracting algorithm E to simulate the $\mathcal{F}$ oracle for D_{mal}. Note that it is easy for D_{hon} to run the extracting algorithm E, which needs the exact same oracles that D_{hon} has access to. The new distinguisher is illustrated in figure 3b.

Say $\mathcal{C}_{\mathcal{G}}$ is a (t_E, q_E) transparent construction. Then if the simulator S_{hon} runs in time $t_{S_{hon}}$ for every query, then S_{mal} runs in time $\mathcal{O}(t_{S_{hon}} \cdot q_E + t_E)$. And if D_{mal} makes $q_{D_{mal}}$ queries and runs in time $t_{D_{mal}}$ then D_{hon} makes at most $\mathcal{O}(q_{D_{mal}} \cdot q_E)$ queries and runs in time $\mathcal{O}(t_{D_{mal}} \cdot t_E)$. ◻

This theorem essentially implies that if one is able to find a transparent construction $C_{\mathcal{G}}$ for an ideal primitive $\mathcal{G}$ and prove its indifferentiability in the honest-but-curious model. This will also imply the general indifferentiability of the construction $C_{\mathcal{G}}$.

3 The Luby-Rackoff Construction

In this section, we will give a construction of an ideal cipher $E : \{0,1\}^{\kappa} \times \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$ from a random oracle $H : \{0,1\}^{*} \rightarrow \{0,1\}^{n}$. Note that it

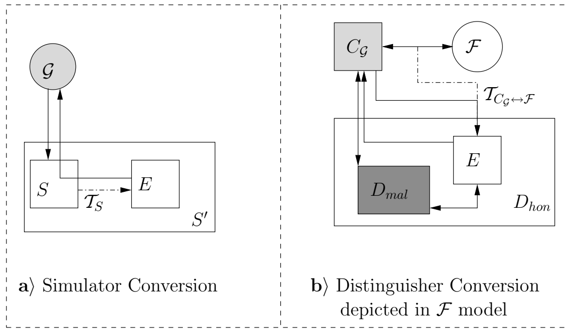

Fig. 3. **a**. Conversion of the simulator S in honest-but-curious model to simulator S' in general indifferentiability.
b. Conversion of the malicious distinguisher D_{mal} into an honest-but-curious distinguisher D_{cur}.

suffices to give a construction C_π of a single random permutation $\pi : \{0,1\}^{2n} \to \{0,1\}^{2n}$ using H. Similar to the ideal cipher oracle, the random permutation oracle π accepts both forward and inverse queries, but it has a key space of cardinality 1. On input $(0, x)$ the oracle outputs $y = \pi(x)$ and on input $(1, y)$ it outputs x such that $\pi(x) = y$. A construction for the ideal cipher E can be easily derived from this random permutation construction by prepending the key of the ideal cipher to every query C_π makes to H.

We will now concentrate on getting an indifferentiable construction of a random permutation from a random oracle, and all our results can be carried over to the ideal cipher model using the technique suggested above.

The Random Permutation Construction. We first note that the constructions in [19, 23] etc. are not necessarily indifferentiable from a random permutation, since all these results are proven in the classical indistinguishability model. Here we will give an indifferentiable construction of *random permutation* (RP) from the *random oracle* (RO) $H : \{0,1\}^* \to \{0,1\}^n$. Similar to [19, 23], our construction is based on multiple rounds of the Feistel permutation. However, our proofs will be in the indifferentiability model. We first formally define a "*k round LR-construction*".

Definition 5 (k round LR-construction). *Given functions $h_i \in F_n : i = 1 \ldots k$, the k round LR-construction $\Psi_{h_1,\ldots,h_k}$ is essentially the composition of k rounds of Feistel permutation, $\Psi_{h_k} \circ \Psi_{h_{k-1}} \circ \ldots \circ \Psi_{h_1}$.*

We will basically use a k round LR-construction (with sufficiently large k) to get a random permutation $\pi : \{0,1\}^{2n} \to \{0,1\}^{2n}$. We will use independent random functions h_i for each round of the k round LR-construction $\Psi_{h_1,\ldots,h_k}$. Note that it is easy to get these independent random functions $h_i \in F_n$ from the random oracle H. These can be simply defined as $h_i(x) = H(\langle i \rangle \parallel x)$ for $i = 1 \ldots k$. Here $\langle i \rangle$ represents the $\log(k)$-bit binary representation of i. The k round LR construction with round functions derived in this fashion is denoted as $\mathcal{C}_{\pi,k}$. We conjecture that for sufficient number of rounds k this is an indifferentiable construction of RP from RO.

Conjecture 1 *For a sufficient number of rounds k, the k round construction $\mathcal{C}_{\pi,k}$ (using a random oracle $H : \{0,1\}^* \to \{0,1\}^n$) is an indifferentiable construction of a random permutation $\pi : \{0,1\}^{2n} \to \{0,1\}^{2n}$.*

Even though we believe this conjecture to hold, we have been unable to prove it formally. However, we will formally show that the k round LR construction is indifferentiable from a random permutation in the honest-but-curious scenario with a sufficient number of rounds k.

3.1 Transparency for $O(\log \lambda)$ Rounds

The question now is how many rounds should suffice to prove indifferentiability in the *honest-but-curious model*? We first show that for upto a logarithmic (in security parameter λ) number of rounds proving indifferentiability of the LR-construction in the honest-but-curious model is no simpler than proving its indifferentiability in general. Recall from section 2 that a *transparent construction* is one for which indifferentiability in the honest-but-curious model implies its indifferentiability in the general model. We prove that for upto a logarithmic (in λ) number of rounds the LR-construction is a transparent construction.

Theorem 2. *The k round LR-construction $\mathcal{C}_{\pi,k}$ is a (t_E, q_E) transparent construction of the random permutation π from random oracle H for number of rounds $k = \mathcal{O}(\log(\lambda))$. The running time t_E and number of queries q_E are both polynomial in the security parameter λ.*

Proof: Consider the k round LR-construction $\mathcal{C}_{\pi,k}$ for number of rounds $k = \mathcal{O}(\log(\lambda))$. We will describe an extracting algorithm E that when given access to $(\mathcal{C}_{\pi,k}, \mathcal{T}_{\mathcal{C}_{\pi,k}\leftrightarrow H})$ can extract the values of $H(\langle i\rangle \parallel x)$ for any $x \in \{0,1\}^n$ and $i = 1 \ldots k$. Note that such an algorithm E will suffice for our purpose. This is because the random oracle output at any other input is never used by the construction $\mathcal{C}_{\pi,k}$. Thus we will assume that E gets inputs of the form $(\langle i\rangle \parallel x)$, and it outputs the value $H(\langle i\rangle \parallel x)$ (or $h_i(x)$). We will describe this algorithm E in an inductive fashion.

- **Input** $(\langle 1\rangle \parallel x)$**:** On this input, E chooses an arbitrary n bit string, R_0. It then assigns $R_1 = x$ and makes the query $\mathcal{C}_{\pi,k}(0, R_0 \parallel R_1)$. This is a forward RP query. In response, it gets the transcript $\mathcal{T}_{\mathcal{C}_{\pi,k}\leftrightarrow H}$, which includes the value $h_1(R_1)$.
- **Input** $(\langle i\rangle \parallel x)$, $i \geq 2$**:** Such round function values are computed recursively.
  - Choose arbitrary $R_0, R_1 \in \{0,1\}^n$, and query $\mathcal{C}_{\pi,k}(0, R_0 \parallel R_1)$. This will give us a random round value R^1_{i-1} and corresponding round function value $h_{i-1}(R^1_{i-1})$.
  - Compute $R^1_{i-2} = h_{i-1}(R^1_{i-1}) \oplus x$. Recursively invoke $E(\langle i-2\rangle \parallel R^1_{i-2})$ to get $h_{i-2}(R^1_{i-2})$.

- Compute $R^1_{i-3} = h_{i-2}(R^1_{i-2}) \oplus R^1_{i-1}$. Recursively invoke $E(\langle i-3 \rangle \parallel R^1_{i-3})$ to get $h_{i-3}(R^1_{i-3})$.
- Continue in this fashion to get $(R^1_{i-4}, h_{i-4}(R^1_{i-4}))$, ... , $(R^1_1, h_1(R^1_1))$.
- Compute $R^1_0 = h_1(R^1_1) \oplus R^1_2$. Now query $\mathcal{C}_{\pi,k}(0, R^1_0 \parallel R^1_1)$. This will give us the round function values $(R^1_i, h_i(R^1_i))$. But $R^1_i = h_{i-1}(R^1_{i-1}) \oplus R^1_{i-2} = x$. Thus we have $h_i(x)$.

For a query $(\langle i \rangle \parallel x)$, let the worst case running time of E be $t_E(i)$ and number of queries be $q_E(i)$. From the above algorithm, we can deduce that $t_E(i) = t_E(i-2) + t_E(i-3) + \ldots + t_E(1) + \mathcal{O}(1)$ and $q_E(i) = q_E(i-2) + q_E(i-3) + \ldots + q_E(1) + \mathcal{O}(1)$. Now one can verify that $t_E(i)$ and $q_E(i)$ are both approximately equal to the i^{th} Fibonacci number. And hence in the worst case $t_E = q_E = \mathcal{O}\left(\phi^k\right)$, where $\phi = \frac{\sqrt{5}+1}{2}$. And thus when $k = \mathcal{O}(\log(\lambda))$, both t_E and q_E are polynomial in the security parameter λ. Hence $\mathcal{C}_{\pi,k}$ is a transparent construction when $k = \mathcal{O}(\log(\lambda))$. $\sqcap\!\!\!\!\sqcup$

Thus one can hope to prove indifferentiability of the LR-construction for $\mathcal{O}(\log(\lambda))$ rounds in the honest-but-curious model, and it will imply the general indifferentiability of the construction. However, there is no indication to suggest that this task might be any easier than the general result.

3.2 Main Result : Equivalence for $\omega(\log \lambda)$ rounds

On the positive side, we prove the indifferentiability of the LR-construction in the honest-but-curious model for a super-logarithmic number of rounds.

Theorem 3. *The k round construction $\mathcal{C}_{\pi,k}$ is $\left(t_D,\ t_S,\ q,\ \mathcal{O}\left((q \cdot k)^4 \cdot 2^{-n}\right)\ \right)$ indifferentiable from a random permutation $\pi : \{0,1\}^{2n} \to \{0,1\}^{2n}$ (with security parameter λ) in the honest-but-curious model for $k = \omega\left(\log(\lambda)\right)$ rounds. t_S, n and q are all polynomial in λ.*

Proof Intuition: The proof of this theorem consists of two parts. First, we will describe the simulator S that fakes the communication between $\mathcal{C}_{\pi,k}$ and H, in the random permutation model. The input to the simulator is either of the form $(0, x)$ (forward π query) or $(1, y)$ (inverse π query), where $x, y \in \{0,1\}^{2n}$. In the random oracle model, if the input $(0, x)$ is given to the construction $\mathcal{C}_{\pi,k}$ in the random oracle model, then $\mathcal{C}_{\pi,k}$ makes queries to the random oracle H and computes the values $R_1 \ldots R_k$ where $R_0 = x|_L$, $R_1 = x|_R$ and $R_i = h_{i-1}(R_{i-1}) \oplus R_{i-2}$ for $i \in \{2, \ldots k+1\}$. Inverse queries $(1, y)$ are handled in a similar fashion, albeit in reverse starting from $R_k = y|_L$ and $R_{k+1} = y|_R$ and computing $R_i = h_{i+1}(R_{i+1}) \oplus R_{i+2}$ for $i \in \{k-1 \ldots 0\}$.

In the random permutation model, the simulator performs essentially the same computation except that it simulates the round functions h_i itself. It maintains a table T_{h_i} for each of the round functions h_i, in which it stores all previously generated round function values. Consider a forward query $(0, x)$, thus $R_0 = x|_L$ and $R_1 = x|_R$. The simulator S generates a fake transcript for this query as follows:

1. First, it forwards this query $(0, x)$ to the random permutation π and gets $y = \pi(x)$. Thus, in our representation of the LR-construction $R_k = y|_L$ and $R_{k+1} = y|_R$.
2. Next, it checks to see if $h_k(R_k)$ is already defined. If so then it checks the tables $T_{h_{k-1}}, T_{h_{k-2}}, \ldots$ and so on to see if there exists a chain of defined values of the form $[R_{i-1} = h_i(R_i) \oplus R_{i+1}]_{i=k\ldots bot}$, where $bot \in \{1, k\}$. If $bot = 1$ then the entire chain is already defined, so it checks to see if the $(R_{bot-1} \parallel R_{bot}) = x$. If so, S returns this sequence of values as the transcript to the distinguisher, otherwise the simulator *exits with failure* since there is no way to define the round function values consistent with π.
3. If $bot > 1$ then it checks to see if similarly there exists a chain of defined round function values going down from $R_0 = x|_L, R_1 = x|_R$. That is, a sequence of round values $[R_{i+1} = h_i(R_i) \oplus R_{i-1}]_{i=1\ldots top}$, where $top \in \{1, k\}$. It then checks to see if $top \geq bot - 2$. If so then it *exits with failure* since it cannot be consistent with both π and its previous responses.
4. If everything goes well until now, then the simulator S starts defining the missing round function values between *top* and *bot*. It defines the function values $h_{top+1}(R_{top+1}) \ldots h_{bot-2}(R_{bot-2})$ at random. It joins the top and bottom chains by defining $h_{bot-1}(R_{bot-1}) = R_{bot} \oplus R_{bot-2}$ and $h_{bot}(R_{bot}) = R_{bot+1} \oplus R_{bot-1}$.
5. After completing the entire chain in this fashion, S sends it to D.

Thus the simulator S simply tries to define all intermediate round function values randomly. However, it first scans to see if part of the chain of round function values is already defined. It does so both starting from top and bottom, and defines the undefined values in the middle at random but making sure that it joins the two partial chain. If it so happens that the two chains are so long that there are no undefined round values left in the middle, then it realizes that it cannot be consistent with both these chains simultaneously and exits with failure.

The next task is to prove the indistinguishability of the random oracle model, with the LR-construction $\mathcal{C}_{\pi,k}$ and the transcript of its communication with the RO H, and the random permutation model, with the random permutation π and the fake transcript generated by the simulator S described above. Our proof consists of a hybrid argument that starts in the random permutation model and through a series of indistinguishable hybrid models it ends up in the random oracle model. The most non-trivial part of the proof consists of the combinatorial lemma 3, which involves counting the number of queries needed by D to induce an inconsistency in the responses of S. This number is shown to be exponential in the number of rounds k, and hence super-polynomial in the security parameter λ when $k = \omega(\log \lambda)$. The formal proof is given below. $\square$

A formal proof of the fact that the simulator described above works is given in appendix A.

3.3 Non-transparency for $\omega(log\lambda)$ rounds

One can deduce from theorem 3 that if the LR-construction with $\omega(\log\lambda)$ rounds is a transparent construction, then it will imply the general indifferentiability of this construction too. Unfortunately, we show that for number of rounds $\omega(\log(\lambda))$ the LR-construction is not a transparent construction.

Theorem 4. *The k round LR-construction $\mathcal{C}_{\pi,k}$ is not a transparent construction of the random permutation π for number of rounds $k = \omega(\log(\lambda))$.*

Proof: Say that we are given an extracting algorithm E that given oracle access to $\mathcal{C}_{\pi,k}$ along with the transcript of the communication between $\mathcal{C}_{\pi,k}$ and the RO H, is supposed to compute H on any input. We will give a query x for which E cannot find $H(x)$ with non-negligible probability.

In the proof of theorem 3, we used a hybrid argument to prove the indistinguishability of $(\mathcal{C}_{\pi,k}, \mathcal{T}_{\mathcal{C}_{\pi,k}\leftrightarrow H})$ from $(\pi, \mathcal{T}_S)$. Recall the hybrid scenario in figure 5b, where we had the simulator S_1 that avoids XOR of any 3 of previously defined round (function) values, and the relaying algorithm $\mathcal{M}_1$ that uses the simulator S_1 to respond to the random permutation queries made by the simulator. By our hybrid argument in the proof of theorem 3, we can see that the random oracle scenario $(\mathcal{C}_{\pi,k}, \mathcal{T}_{\mathcal{C}_{\pi,k}\leftrightarrow H})$ is also indistinguishable from this hybrid scenario $(\mathcal{M}_1, \mathcal{T}_{S_1})$.

Coming back to our current proof, if we give the extracting algorithm E access to $(\mathcal{M}_1, \mathcal{T}_{S_1})$, then it should be able to compute the output of any of the round functions simulated by S_1 on any input, just as it does in the random oracle model. If this is not the case, then we can use the extracting algorithm E to design a distinguisher that can tell apart the random oracle model from this hybrid model with high probability. Let us denote the round functions simulated by S_1 as $h_1 \dots h_k$ and the corresponding round values as $R_0, \dots, R_{k+1}$.

We will ask the extracting algorithm E to compute $h_{\frac{k}{2}}(x)$. Say E finds out $h_{\frac{k}{2}}(x)$ in query number m, which can be assumed to be a forward query without loss of generality. Denote the round values in query number m as $R_0^{(m)}, \dots, R_{k+1}^{(m)}$. We can deduce that $R_{(k/2)}^{(m)} = x$ since it is in this query that E finds the values $h_{\frac{k}{2}}(x)$. Now if the round value $R_{(k/2)-1}^{(m)}$ is a new round value then $h_{\frac{k}{2}-1}(R_{(k/2)-1}^{(m)})$ would have been assigned a random value and $h_{\frac{k}{2}-1}(R_{(k/2)-1}^{(m)}) \oplus R_{(k/2)-2}^{(m)}$ would have been equal to x with only a negligible probability. So it must have been the case that $R_{(k/2)-1}^{(m)}$ was defined in some query prior to query number m. We can make similar deductions to show that all the round values $R_0^{(m)}, \dots, R_{(k/2)-2}^{(m)}$ were also defined in queries previous to the m^{th} query.

After this the proof of the theorem follows in pretty much the same way as the combinatorial lemma 3. We show that the extracting algorithm must have already made a $\phi^{\frac{k}{4}}$ (for $\phi = \frac{\sqrt{5}+1}{2}$) queries prior to the m^{th} query. For a super-logarithmic number of rounds k, this is super-polynomial in the security parameter λ. $\sqcup\!\sqcap$

3.4 Negative Results for Constant Rounds

Finally, we mention that one does need to use sufficient number of rounds of the Feistel permutation in the construction, to have any hope of proving it indifferentiable. Coron [7] showed that for less than 6 rounds the LR-construction is not indifferentiable from a random permutation.

Theorem 5 ([7]). *Let $\mathcal{C}_{\pi,k}$ be the k round LR-construction of a random permutation π, with number of rounds $k < 6$. Then there is an efficient distinguisher D such that for any simulator S, D can distinguish the oracle pair $(\mathcal{C}_{\pi,k}, H)$ and (π, S) with non-negligible probability.*

It is easy to see that the construction $(\mathcal{C}_{\pi,k}, H)$ cannot work for $k < 4$, since in this case it does not even satisfy the classical indistinguishability definition [19]. Coron [7] gave attacks on 4 and 5 round LR-constructions in the indifferentiability scenario. We give an attack on the 4 round LR construction in appendix B for illustration.
This theorem also implies that indifferentiability (even in the honest-but-curious model) is strictly stronger than classical indistinguishability. This is because the LR-construction with 4 rounds or more is known to satisfy the latter [19]. Thus we can derive the following corollary from theorem 5.

Corollary 1. *A 4 round LR-construction is indistinguishable , but not indifferentiable, from a random permutation (even in the honest-but-curious model).*

4 Conclusions and Future work

In this paper, we have shown that the Luby-Rackoff construction with a super-logarithmic number of rounds can be used to instantiate the ideal block cipher in any honest-but-curious cryptosystem. We have also proved that improving this result to upto a logarithmic number of rounds will imply that this construction is indifferentiable from the ideal cipher in general. The main question that still remains unanswered is whether the Luby-Rackoff construction is indifferentiable from the ideal cipher in general.

Acknowledgements. We would like to thank Jean-Sébastien Coron and Joel Spencer for useful discussions.

References

1. J. Black, *The Ideal-Cipher Model, Revisited: An Uninstantiable Blockcipher-Based Hash Function*, *eprint 2005/210*, (2005).
2. M. Bellare, A. Boldyreva and A. Palacio. *An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem*, To appear in *Proccedings of Eurocrypt* (2004).
3. M. Bellare, and P. Rogaway, *Random oracles are practical: A paradigm for designing efficient protocols*, In *Proceedings of the 1st ACM Conference on Computer and Communications Security* (1993), 62 -73.

4. M. Bellare and P. Rogaway, *The exact security of digital signatures - How to sign with RSA and Rabin*. Proceedings of Eurocrypt'96, LNCS vol. 1070, Springer-Verlag, 1996, pp. 399-416.
5. J. Black, P. Rogaway, T. Shrimpton, *Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV*, in Advances in Cryptology - CRYPTO 2002, California, USA.
6. M. Bellare and P. Rogaway, *Optimal Asymmetric Encryption*, Proceedings of Eurocrypt'94, LNCS vol. 950, Springer-Verlag, 1994, pp. 92–111.
7. J.-S. Coron, *personal communication*.
8. J.-S. Coron, Y. Dodis, C. Malinaud and P. Puniya, *Merkle-Dåmgard Revisited: How to Construct a Hash Function*, In *Advances in Cryptology - Crypto 2005 Proceedings* (2005), 430 -448.
9. J.-S. Coron, A. Joux, and D. Pointcheval, *Equivalence Between the Random Oracle Model and the Random Cipher Model*, *Dagstuhl Seminar 02391: Cryptography*, (2002).
10. R. Canetti, O. Goldreich, and S. Halevi, *The random oracle methodology, revisited*, In *Proceedings of the 30th ACM Symposium on the Theory of Computing* (1998) , ACM Press, pp. 209 -218.
11. R. Canetti, O. Goldreich, and S. Halevi, *On the random-oracle methodology as applied to length-restricted signature schemes*, In *First Theory of Cryptography Conference* (2004).
12. I. Dåmgard, *A Design Principle for Hash Functions*, In Crypto '89, pages 416-427, 1989. LNCS No. 435.
13. A. Desai, *The security of all-or-nothing encryption: Protecting against exhaustive key search*, In *Advances in Cryptology - Crypto'00* (2000), LNCS vol. 1880, Springer-Verlag.
14. S. Even, and Y. Mansour, *A construction of a cipher from a single pseudorandom permutation*, In *Advances in Cryptology - ASIACRYPT'91* (1992), LNCS vol. 739, Springer-Verlag, pp. 210 -224.
15. A. Fiat, and A. Shamir, *How to prove yourself: Practical solutions to identification and signature problems*, In *Advances in Cryptology - Crypto'86* (1986), Lecture Notes in Computer Science, Springer-Verlag, pp. 186 -194.
16. S. Goldwasser and Y. Tauman. *On the (In)security of the Fiat-Shamir Paradigm*, In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science* (2003), 102-114.
17. E. Jaulmes, A. Joux, and F. Valette, *On the security of randomized CBC-MAC beyond the birthday paradox limit: A new construction*, In *Fast Software Encryption (FSE 2002)* (2002), vol. 2365 of Lecture Notes in Computer Science, Springer-Verlag, pp. 237 -251.
18. J. Kilian, and P. Rogaway, *How to protect DES against exhaustive key search (An analysis of DESX)*, *Journal of Cryptology 14*, 1 (2001), 17 -35.
19. M. Luby and C. Rackoff, *How to construct pseudo-random permutations from pseudo-random functions*, SIAM J. Comput., Vol. 17, No. 2, April 1988.
20. U. Maurer, R. Renner, and C. Holenstein, *Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology*, Theory of Cryptography - TCC 2004, Lecture Notes in Computer Science, Springer-Verlag, vol. 2951, pp. 21-39, Feb 2004.
21. R. Merkle, *One way hash functions and DES*, in *Advances in Cryptology, Proc. Crypto'89, LNCS 435*, G. Brassard, Ed., Springer-Verlag, 1990, pp. 428-446.

22. J. B. Nielsen. *Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-Committing Encryption Case*, In *Advances in Cryptology - Crypto 2002 Proceedings* (2002), 111-126
23. M. Naor and O. Reingold, *On the construction of pseudo-random permutations: Luby-Rackoff revisited*, J. of Cryptology, vol 12, 1999, pp. 29-66.
24. D. Pointcheval, and J. Stern, *Security proofs for signature schemes*, In *Advances in Cryptology - Eurocrypt 1996 proceedings*, 387 -398.
25. C.-P. Schnorr, *Efficient signature generation by smart cards*, In *Journal of Cryptology 4*, 3 (1991), 161 -174.
26. R. Winternitz, *A secure one-way hash function built from DES*, in Proceedings of the IEEE Symposium on Information Security and Privacy, pages 88-90. IEEE Press, 1984.

A Formal Proof of Indifferentiability

Now we will prove that when the simulator S described in theorem 3 above is used in the indifferentiability game, then any distinguisher D that makes at most q queries to its oracles has only a negligible distinguishing advantage. Here q and n (the output length of H) are both polynomial functions of the security parameter λ, while the number of rounds in the LR construction is $k = \omega(\log(\lambda))$. As we mentioned, our proof proceeds via a hybrid argument.

Hiding the random permutation π: Let us start in the random permutation scenario. Here the distinguisher has oracle access to π and the simulator S. Our first modification is to prevent D from directly accessing π, by replacing it with a simple relaying algorithm $\mathcal{M}$ that acts as an interface to π. When $\mathcal{M}$ gets a query from the distinguisher, it simply relays this query to the random permutation π and sends back the response of π. In this new scenario, the distinguisher has oracle access to $\mathcal{M}^\pi$ and S^π (see figure 5a). Since we have made no real change from the point of view of the distinguisher, we have $Pr[D^{(\pi,\mathcal{T}_{S^\pi})} = 1] = Pr[D^{(\mathcal{M}^\pi,\mathcal{T}_{S^\pi})} = 1]$.

Bounding out the "bad events": Now we will modify the simulator S, so that it never outputs certain types of collisions that will affect our analysis later. Recall that the simulator S needs to define the round function values $h_1(R_1)\dots h_k(R_k)$ in order to generate the transcript $\mathcal{T}_S$ for every query made to it. And S tries to assign random values to $h_i(R_i)$ for any new R_i.

Now we introduce a slightly modified simulator S_1 that is essentially the same as S except that it chooses round function values more carefully. Let us first fix a little notation. We will number the queries made to the simulator in the order they are made, query number 1 followed by 2 and so on. And for the m^{th} query made to the simulator, we will label its round values as $R_0^{(m)}, R_1^{(m)}, \dots, R_k^{(m)}, R_{k+1}^{(m)}$.

Assume for now that query number m is a forward query. When assigning a new round function value $h_i(R_i^{(m)})$ in this query, the distinguisher makes sure that $h_i(R_i^{(m)})$ is cannot be represented as an *XOR of upto three previously defined values*. This includes all values $R_j^{(\ell)}$ or $h(R_j^{(\ell)})$ for $\ell < m$, $j \in$

$\{0, k+1\}$ and all values $R_j^{(m)}$ or $h_j(R_j^{(m)})$ for $j < i$ (and $j > i$ for an inverse query m). More formally, S_1 assigns a value $h_i(R_i^{(m)})$ for the m^{th} query (a forward query) that does not satisfy the following equality for values $x_1, x_2, x_3 \in \{R_{j_1}^{(\ell)}, h_{j_1}(R_{j_1}^{(\ell)}), R_{j_2}^{(m)}, h_{j_2}(R_{j_2}^{(m)}) \mid \ell < m\ ,\ j_1 \leq k+1,\ j_2 < i\}$

$$h_i(R_i^{(m)}) = x_1 \ or \ (x_1 \oplus x_2) \ or \ (x_1 \oplus x_2 \oplus x_3)$$

The distinguisher cannot tell if it has oracle access to $(\mathcal{M}, S)$ or $(\mathcal{M}, S_1)$ unless the old simulator S outputs a round function value that satisfies one of the above equalities. Let us denote this event by B_1. Hence for any distinguisher D making q queries,

$$\left| Pr\left[D^{(\mathcal{M}^\pi, \mathcal{T}_{S^\pi})} = 1\right] - Pr\left[D^{(\mathcal{M}^\pi, \mathcal{T}_{S_1^\pi})} = 1\right]\right| \leq Pr\left[B_1\right]$$

We can bound the probability of B_1 occurring by noticing that for randomly assigned round function values, $Pr\left[B_1\right] = \mathcal{O}\left(\frac{(q \cdot k)^4}{2^n}\right)$. This can be derived by using the birthday paradox to bound the probability that any XOR of upto 4 round (or round function) values is 0^n.

Transferring Control to the Simulator: Next we will modify the relaying algorithm $\mathcal{M}$ so that it does not simply act as a channel between the distinguisher and π. The new relaying algorithm, which we will call $\mathcal{M}_1$, responds to the π queries by making the same queries to the simulator S_1 and computing $\pi(x)$ (or $\pi^{-1}(y)$) from the responses of S_1 (see figure 5b).

To illustrate this point, say $\mathcal{M}_1$ gets a query $(0, x)$ from the distinguisher D (that is, a forward query to π). Then $\mathcal{M}_1$ forwards this query to S_1, which in turn gets $y = \pi(x)$ from the random permutation and constructs a fake transcript $\mathcal{T}_{S_1}(0, x)$ (or round values $R_0 = x|_L, R_1 = x|_R, \ldots, R_{k+1}$). If all goes well this transcript is consistent with π. The simulator sends this transcript $\mathcal{T}_{S_1}(0, x)$ to $\mathcal{M}_1$, which can compute $\pi(x)$ from $\mathcal{T}_{S_1}$ and send it to the distinguisher D with this value. Inverse queries $1, y)$ are handled in a similar fashion.

From the view of D, everything in this scenario is same as in the previous one unless the simulator S_1 exits with failure on some query made by $\mathcal{M}_1$. This happens if and only if S_1 fails to be consistent with the random permutation π on some query. We claim that if the number of queries q made by the distinguisher D is polynomial in the security parameter λ then the simulator S_1 is always consistent with π.

Lemma 3. For a polynomial number of queries q made to the simulator S_1, the responses of the simulator are always consistent with the random permutation π.

Proof: Say query number m is the first time S_1 is inconsistent with π. Without loss of generality assume this to be a forward query $(0, x)$, with $\pi(x) = y$. Thus $R_0^{(m)} = x|_L, R_1^{(m)} = x|_R$ and $R_k^{(m)} = y|_L, R_{k+1}^{(m)} = y|_R$. Since S_1 is inconsistent on this query, there exist partial round value chains, $R_0^{(m)}, R_1^{(m)} \ldots R_{top}^{(m)}, R_{top+1}^{(m)}$ and

$R_{bot-1}^{(m)}, R_{bot}^{(m)} \ldots R_k^{(m)}, R_{k+1}^{(m)}$ with $top \geq bot - 2$. But in this case either ($top \geq \frac{k}{2}$) or ($bot \leq \frac{k}{2} + 1$). That is, at least one of these two partial chains consists of more than $\frac{k}{2}$ defined round function values. Without loss of generality, assume that the $top \geq \frac{k}{2}$. Thus all round function values $h_1(R_1^{(m)}) \ldots h_{top}(R_{top}^{(m)})$ were defined before query number m was made. We will look at the queries where each of these round function values was defined for the first time. For any round value $R_i^{(j)}$, we denote by $first(R_i^{(j)})$ the query number where the round function value $h_i(R_i^{(j)})$ was first defined. Thus if $R_i^{(j)}$ is a new round value that appeared in query number j itself, then $first(R_i^{(j)}) = j$ otherwise $first(R_i^{(j)}) < j$. We can thus say that for $i = 1 \ldots top$, it is the case that $first(R_i^{(m)}) < m$.

Now consider any three consecutive round values $R_{i-1}^{(m)}$, $R_i^{(m)}$ and $R_{i+1}^{(m)}$ for $i \in \{2 \ldots top - 1\}$. Let $first(R_{i-1}^{(m)}) = \ell_{i-1}$, $first(R_i^{(m)}) = \ell_i$ and $first(R_{i+1}^{(m)}) = \ell_{i+1}$ ($\ell_{i-1}, \ell_i, \ell_{i+1} < m$). We wish to analyze the order of the queries ℓ_{i-1}, ℓ_i and ℓ_{i+1}. First, note that $\ell_{i-1} \neq \ell_i$ and $\ell_i \neq \ell_{i+1}$. Either case would imply that the ${\ell_i}^{th}$ query is the same as the m^{th} query, and the inconsistency should have occurred there itself. Let us now look at the possible orders between ℓ_{i-1}, ℓ_i and ℓ_{i+1}.

1. $\underline{(\ell_i > \ell_{i-1} \geq \ell_{i+1}) \text{ or } (\ell_i > \ell_{i+1} > \ell_{i-1})}$: That is, query number ℓ_i occurs after ℓ_{i-1} and ℓ_{i+1}. We know that $h_i(R_i^{(m)}) = R_{i-1}^{(m)} \oplus R_{i+1}^{(m)}$ and hence $h_i(R_i^{(\ell_i)}) = R_{i-1}^{(\ell_{i-1})} \oplus R_{i+1}^{(\ell_{i+1})}$. But the round values $R_{i-1}^{(\ell_{i-1})}$ and $R_{i+1}^{(\ell_{i+1})}$ already exist when $h_i(R_i^{(\ell_i)})$ was defined for the first time in the ${\ell_i}^{th}$ query. And since the simulator S_1 avoids such an XOR collision, this order is *impossible*.
2. $\underline{(\ell_{i-1} > \ell_i > \ell_{i+1}) \text{ or } (\ell_{i-1} < \ell_i < \ell_{i+1})}$: These strictly increasing/decreasing orderings are *possible*.
3. $\underline{(\ell_i < \ell_{i-1} < \ell_{i+1}) \text{ or } (\ell_i < \ell_{i+1} < \ell_{i-1})}$: Here the ${\ell_i}^{th}$ query comes before both the ${\ell_{i-1}}^{th}$ and ${\ell_{i+1}}^{th}$ queries. These orders are *possible*.
4. $\underline{(\ell_i < \ell_{i-1} = \ell_{i+1})}$: This is the same as above, except that the $\ell_{i-1} = \ell_{i+1}$. In this case, a short calculation gives that $h_i(R_i^{(\ell_{i-1})}) = h_i(R_i^{(\ell_i)})$, where $R_i^{(\ell_{i-1})} \neq R_i^{(\ell_i)}$. And since $R_i^{(\ell_{i-1})}$ exists before $h_i(R_i^{(\ell_i)})$ is defined, this order is *impossible*.

Thus we know that the possible orderings of the queries for any three consecutive round values are the configurations 2 and 3. Now we can apply the same to all the queries $first(R_1^{(m)}) = \ell_1$, $first(R_2^{(m)}) = \ell_2, \ldots, first(R_{top}^{(m)}) = \ell_{top}$, considering each triple of consecutive round values separately and then combining of these orderings together. Using this, we obtain that there is a $j \in \{1, k\}$ such that ($\ell_1 > \ell_2 > \ldots > \ell_j$) and ($\ell_j < \ell_{j+1} < \ldots < \ell_{top}$). That is, the query numbers $\ell_1 \ldots \ell_{top}$ are strictly decreasing until some ℓ_j and strictly increasing after that. One can verify that any other configuration will involve one of the "impossible" triple orderings 1 or 4.

Now we will look for more structure in these queries. If $j \geq \frac{top}{2}$, then we will analyze the decreasing sequence of queries $\ell_1 \ldots \ell_j$, otherwise we will analyze the increasing sequence of queries $\ell_j \ldots \ell_{top}$. Without loss of generality, assume that

$j \geq \frac{top}{2}$; the case $j < \frac{top}{2}$ is symmetrical. Since we earlier derived that $top \geq \frac{k}{2}$, we can also deduce that $j \geq \frac{k}{4}$.

Now we will show that these queries and others that led to the inconsistency in the m^{th} query form a *Fibonacci tree* of depth j (which we know is $\geq \frac{k}{4}$). Each node of the Fibonacci tree corresponds to a different query, with m^{th} query at the root of the tree. This would imply that m is at least as large as the number of nodes in a *Fibonacci tree* of depth $\frac{k}{4}$. But since we know that $k = \omega(\log(\lambda))$ it also holds that m is superpolynomial in the security parameter λ. In turn, this implies that the simulator S_1 is always consistent with the random permutation for any polynomial number of queries.

The queries from $\ell_1 \dots \ell_j$ form the first level of the *Fibonacci tree* which we will describe. To see this structure more explicitly, we will now move from the m^{th} query to these first level queries. Consider any three consecutive queries in this ordering, $\ell_i, \ell_{i+1}, \ell_{i+2}$ (recall $\ell_i > \ell_{i+1} > \ell_{i+2}$). Let us look at the ${\ell_i}^{th}$ query. This query could be a forward or inverse query. For now we assume that it is a forward query. As it will turn out, if this is an inverse query then the *Fibonacci tree* of queries would be even larger, and so will the number of queries needed. We know that $R_i^{(\ell_i)} (= R_i^{(m)})$ is a new round value in this query. Consider the round function value $h_{i-1}(R_{i-1}^{(\ell_i)})$. Since $R_i^{(\ell_i)} = R_i^{(m)}$, we can deduce that

$$h_{i-1}\left(R_{i-1}^{(\ell_i)}\right) = R_{i-2}^{(\ell_i)} \oplus h_{i+1}\left(R_{i+1}^{(\ell_{i+1})}\right) \oplus R_{i+2}^{(\ell_{i+2})}$$

Note that the ${\ell_{i+1}}^{th}$ and ${\ell_{i+2}}^{th}$ queries were made before query number ℓ_i. And since the ${\ell_i}^{th}$ query is a forward one, $R_{i-2}^{(\ell_i)}$ is defined before $R_{i-1}^{(\ell_i)}$. Now if $R_{i-1}^{(\ell_i)}$ is a new round value then the simulator S_1 would have avoided the above XOR representation. Thus $h_{i-1}(R_{i-1}^{(\ell_i)})$ was already defined before the ${\ell_i}^{th}$ query. Using similar analysis, one can also deduce that the round function values $h_1(R_1^{(\ell_i)}) \dots h_{i-2}(R_{i-2}^{(\ell_i)})$ also had to be defined prior to the ${\ell_i}^{th}$ query.

Let $first(R_1^{(\ell_i)}) = \mathsf{b}_1, \dots, first(R_{i-1}^{(\ell_i)}) = \mathsf{b}_{i-1}$. Consider the queries b_{i-1} and b_{i-2}. Let us see in what order these queries could have occurred. We know that queries b_{i-1} and b_{i-2} were both made before the ${\ell_i}^{th}$ query. We also know that the ${\ell_{i+1}}^{th}$ query was also made before ${\ell_i}^{th}$ query. First note that $\mathsf{b}_{i-1} \neq \mathsf{b}_{i-2}$, since otherwise the ${\mathsf{b}_{i-1}}^{th}$ query would the same as query number ℓ_i, which is not possible since $R_i^{(\ell_i)}$ is a new round values in the ${\ell_i}^{th}$ query.

1. $\underline{\mathsf{b}_{i-2} < \mathsf{b}_{i-1} \leq \ell_{i+1} \text{ or } \mathsf{b}_{i-1} < \mathsf{b}_{i-2} \leq \ell_{i+1}}$: A short calculation in this case gives $h_{i+1}(R_{i+1}^{(\ell_{i+1})}) = R_{i+2}^{(\ell_{i+2})} \oplus R_{i-2}^{(\mathsf{b}_{i-2})} \oplus h_{i-1}(R_{i-1}^{(\mathsf{b}_{i-1})})$. Since all 3 of these round (function) values existed before $h_{i+1}(R_{i+1}^{(\ell_{i+1})})$ was defined, the simulator S_1 would have avoided their XOR. Hence these orderings are *impossible*.
2. $\underline{\mathsf{b}_{i-2} \leq \ell_{i+1} < \mathsf{b}_{i-1} \text{ or } \ell_{i+1} < \mathsf{b}_{i-2} < \mathsf{b}_{i-1}}$: These orderings are *impossible* since here we can make a similar argument for $h_{i-1}(R_{i-1}^{(\mathsf{b}_{i-1})})$.
3. $\underline{\mathsf{b}_{i-1} \leq \ell_{i+1} < \mathsf{b}_{i-2}}$: This ordering is *possible*.
4. $\underline{\ell_{i+1} < \mathsf{b}_{i-1} < \mathsf{b}_{i-2}}$: This ordering is also *possible*.

We note here a couple of things about these possible orderings before we move on. First, query b_{i-1} could have only been made before query b_{i-2}. Secondly, query b_{i-2} could not have been made before the query ℓ_{i+1}. Now starting with this ordering defined between queries b_{i-1} and b_{i-2}, we can deduce the order in which queries $\mathsf{b}_1 \ldots \mathsf{b}_{i-3}$ could have been made. The analysis of this will be pretty much the same as that for $\ell_1 \ldots \ell_{top}$, with one major difference. Here the only possible order amongst $\mathsf{b}_1 \ldots \mathsf{b}_{i-1}$ we will get will be a descending order $\mathsf{b}_1 > \mathsf{b}_2 > \ldots > \mathsf{b}_{i-1}$. That is query b_1 was made before b_2 which was made before b_3 and so on. This happens because we were able to establish a strict order between b_{i-2} and b_{i-1}, which was not the case for ℓ_{top-1} and ℓ_{top}. Thus the $i-1$ queries, $\mathsf{b}_1 \ldots \mathsf{b}_{i-1}$, had to be made in strict decreasing order. This fact turns out to be really crucial since we do not lose half of the queries at this level of the "Query tree", as we did in the case of query number m.

Thus for each of the queries ℓ_i at the first level, we have at least $i-2$ queries that lie strictly in between ℓ_i and ℓ_{i+1}. Note that the same counting method can be extended to the b_i queries to show that there are $i-2$ queries strictly in between b_i and b_{i+1}, and so on. This query structure takes the shape of a *Fibonacci tree*. Since queries at any level lie strictly in between two consecutive parent level queries, it turns out that each of the queries in the tree is, in fact, different! An example this query structure is shown in figure 4.

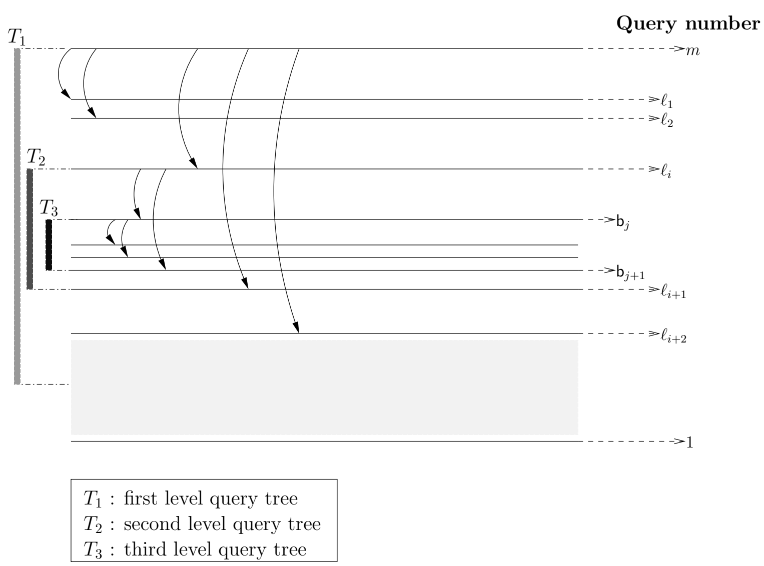

Fig. 4. An example of a "Fibonacci tree" formed by queries (showing three levels)

Let $T(i)$ represents the number of queries in a "query tree" starting with $R_i^{(m)}$ (thus $T(1) = 1$). From the structure of the "query tree", we can compute that $T(i) = T(i-1) + T(i-2)$. But this is exactly the expression for the i^{th} *Fibonacci number*. We will not recompute this expression here and just state that $T(i) = \mathcal{O}\left(\phi^i\right)$ where $\phi = \frac{\sqrt{5}+1}{2}$. And thus if an inconsistency occurs in query number m, then $m = \mathcal{O}\left(T\left(\frac{k}{4}\right)\right) = \mathcal{O}\left(\phi^{\frac{k}{4}}\right)$, which is superpolynomial in the security parameter λ, if $k = \omega(\log(\lambda))$. $\boxminus$

Thus for any distinguisher D that makes q queries ($q = poly(\lambda)$), it is the case that $Pr[D^{(\mathcal{M}^{\pi}, \mathcal{T}_{S_1^{\pi}})}] = Pr[D^{(\mathcal{M}^{\mathcal{T}_{S_1^{\pi}}}, \mathcal{T}_{S_1^{\pi}})}]$.

Removing the Random Permutation π: Until now, all queries are forced to be consistent with π. Now we will modify the simulator S_1 and get closer to the actual random oracle scenario. The new simulator, which we shall denote by S_2, does not attempt to output transcripts consistent with π. As before it implements the k round LR-construction with randomly assigned internal round functions. But now it also implements the last (or first) couple of round functions h_{k-1}, h_k (or h_2, h_1) with randomly chosen values (see figure 5c).

To illustrate this, when the new simulator S_2 gets a forward query $(0, x)$. It computes $R_0 = x|_L, R_1 = x|_R$ and assigns random values to $h_1(R_1), \ldots, h_k(R_k)$. It then sends the round values $R_0, \ldots, R_{k+1}$ as the transcript for the query $(0, x)$. Inverse queries are handled in a symmetrical fashion. The relaying algorithm, $\mathcal{M}_1$, as before uses these transcripts to compute its responses to D's queries.

Note that the distinguisher cannot tell this scenario apart from the previous scenario, unless

- the new simulator S_2 violates the XOR constraint satisfied by S_1. We call this event B_3.
- the old simulator S_1 exits with failure. We call this event B_4.

Lemma 3 implies that the event B_4 does not happen for any distinguisher D that makes a polynomial number of queries. Thus for any distinguisher D making at most a polynomial number of queries q,

$$\left| Pr\left[D^{(\mathcal{M}^{\mathcal{T}_{S_1^{\pi}}}, \mathcal{T}_{S_1^{\pi}})}\right] - Pr\left[D^{(\mathcal{M}^{\mathcal{T}_{S_2}}, \mathcal{T}_{S_2})}\right]\right| \leq Pr\left[B_3\right] = \mathcal{O}\left(\frac{(q.k)^4}{2^n}\right)$$

Onto the Random Oracle Model: Note that the previous scenario is essentially the same as the random oracle scenario, since all round function values chosen by S_2 are random. Therefore for any distinguisher D (figure 5d), we have $Pr[D^{(\mathcal{M}^{\mathcal{T}_{S_2}}, \mathcal{T}_{S_2})}] = Pr[D^{(\mathcal{C}^H_{\pi,k}, \mathcal{T}_{\mathcal{C}_{\pi,k} \leftrightarrow H})} = 1]$.

Combining all the above hybrids, for any distinguisher D that makes at most q queries,

$$\left| Pr\left[D^{(\mathcal{C}^H_{\pi,k}, \mathcal{T}_{\mathcal{C}_{\pi,k} \leftrightarrow H})} = 1\right] - Pr\left[(D^{\pi, \mathcal{T}_{S^{\pi}})} = 1\right]\right| < \mathcal{O}\left(\frac{(q \cdot k)^4}{2^n}\right)$$

Here q and n are polynomial in the security parameter λ, and $k = \omega(\log(\lambda))$. In fact, with a slightly more carefully designed simulator S_1 that avoids an XOR of specific round (function) values, one gets that the distinguishing advantage of D is $\mathcal{O}\left(\frac{q^4}{2^n}\right)$

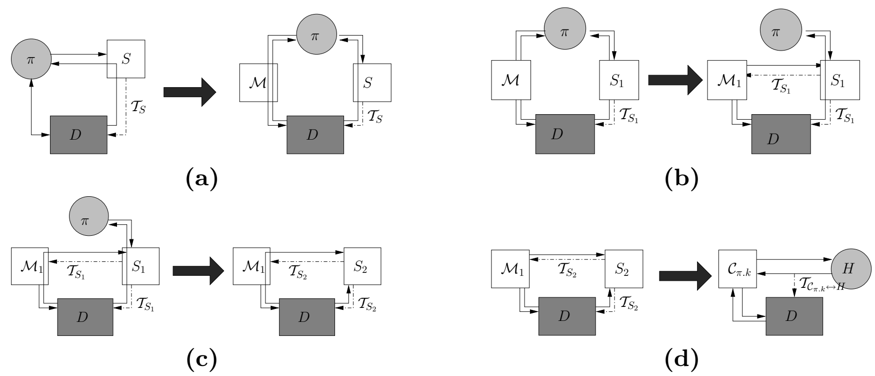

Fig. 5. Overall Game Structure

B Attack on 4 round LR-construction

We will represent the round values of the construction $\mathcal{C}_{\pi,4}$ as $R_0, R_1 \ldots R_4, R_5$, such that $\mathcal{C}_{\pi,4}(R_0 \parallel R_1) = (R_4 \parallel R_5)$. And the round functions will be denoted as $h_1, \ldots, h_4$. Now consider any simulator S for which we get the two scenarios: $(\mathcal{C}_{\pi,4}, H)$ and (π, S). We will design a distinguisher D that distinguishes these two with high probability for any simulator S.

The distinguisher D essentially forces the simulator to satisfy a constraint that holds with very low probability for an RP π. On the other hand, it always holds for the LR-construction $\mathcal{C}_{\pi,4}$. The algorithm of D is as follows:

1. Choose 3 arbitrary n bit strings, $R_2,\ R_2',\ R_3$.
2. Query the random oracle H to get $h_2(R_2)$, $h_2(R_2')$ and $h_3(R_3)$, in this order.
3. Compute $R_1 = h_2(R_2) \oplus R_3$ and $R_1' = h_2(R_2') \oplus R_3$.
4. Query the random oracle to get $h_1(R_1)$ and $h_1(R_1')$. Compute $R_0 = h_1(R_1) \oplus R_2$ and $R_0' = h_1(R_1') \oplus R_2$.
5. Query the random permutation on $R_0 \parallel R_1$ and $R_0' \parallel R_1'$ to get the values $R_4 \parallel R_5$ and $R_4' \parallel R_5'$, respectively.
6. Check if $R_4 \oplus R_4' = R_2 \oplus R_2'$. If so, then output 1 else output 0

Note that the values R_2 and R_2' were queried upon before R_3. Hence the round values R_1 and R_1' are completely arbitrary round values controlled by the distinguisher. The distinguisher D always outputs 1 when given access to the construction $\mathcal{C}_{\pi,4}$. But when given access to the random permutation, the simulator S will need to find $h_1(R_1)$ and $h_1(R_1')$ that satisfy the constraint:

$$\pi((h_1(R_1) \oplus R_2) \parallel R_1)|_L \oplus \pi((h_1(R_1') \oplus R_2') \parallel R_1')|_L = R_2 \oplus R_2'$$

In this equation R_1, R_1', R_2 and R_2' are all effectively chosen by the distinguisher. Hence no efficient simulator can find two round function values $h_1(R_1)$ and $h_1(R_1')$ that satisfy the above constraint with non-negligible probability for a random permutation π.