New Design Criteria for Hash Functions and Block Ciphers.

by

Prashant Puniya

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

September 2007

Yevgeniy Dodis

Abstract

Cryptographic primitives, such as hash functions and block ciphers, are integral components in several practical cryptographic schemes. In order to prove security of these schemes, a variety of security assumptions are made on the underlying hash function or block cipher, such as collision-resistance, pseudorandomness etc. In fact, such assumptions are often made without much regard for the actual constructions of these primitives. In this thesis, we address this problem and suggest new, and possibly better, design criteria for hash functions and block ciphers.

We start by analyzing the design criteria underlying hash functions. The usual design principle here involves a two-step procedure: First, come up with a heuristically-designed and "hopefully strong" fixed-length input construction (i.e. *the compression function*), then use a standard domain extension technique, usually the *cascade construction* (see figure 3.2), to get a construction that works for variable-length inputs. We investigate this design principle from two perspectives:

(a) To instantiate the *Random Oracle*. We suggest modifications to existing constructions that make the resulting construction secure as a random oracle, with appropriate assumptions on the underlying compression function.

(b) In general, we look for "black-box" fixes to existing hash functions to get secure constructions for each of the common security notions required of hash functions. We also give suggestions for appropriate modes for using existing hash functions along these lines.

We next move on to discuss the *Feistel network*, which is used in the design of several popular block ciphers such as DES, Triple-DES, Blowfish etc. Currently, the celebrated result of Luby-Rackoff [47] (and further extensions) is regarded as the theoretical basis for using this construction in block cipher design, where it was shown that a four-round Feistel network is a *(strong) pseudorandom permutation* (PRP) if the round functions are independent *pseudorandom functions* (PRFs). We study the Feistel network from two different perspectives:

(a) Is there a weaker security notion for round functions, than pseudorandomness, that suffices to prove security of the Feistel network?

(b) Can the Feistel network satisfy a much stronger security notion, i.e. security as an ideal cipher, under appropriate assumptions on the round functions?

We give a positive answer to the first question and a partial positive answer to the second question. In the process, we undertake a combinatorial study of the Feistel network, that might be useful in other scenarios as well. We provide several practical applications of our results for the Feistel network.

Contents

List of Figures

Chapter 1

Introduction

Cryptographic primitives, such as hash functions and block ciphers, are integral components in the design of practical cryptographic schemes. Often the use of such primitives makes the task of coming up with secure and efficient cryptosystems much easier, as compared to designing such systems from scratch based on complexity-theoretic assumptions. The usual design procedure involves coming up with a proposed construction that uses an abstract function/permutation family. The construction is then proven secure by making an appropriate assumption on the function/permutation family. For instance, assuming the function family to be collision-resistant or assuming the permutations to be *pseudorandom permutations*. In practice, these functions (resp. permutation) families are instantiated with actual hash functions (resp. block ciphers), in the hope that these constructions will satisfy the required security notion.

Hence, depending on the requirements of cryptographic schemes these primitives may need to satisfy a variety of security notions. For this reason, the notion of a “secure” hash function or a “secure” block cipher is a little fuzzy, at best. In this thesis, we attempt to come up with new and possibly better design criteria for these primitives.

1.1 Hash Functions

The most common way of constructing a hash functions consists of two steps. First, one constructs a *compression function* $f : \{0,1\}^m \rightarrow \{0,1\}^n$ from scratch, or using a block cipher. Then one uses an iterative technique such as the *Cascade construction* (see figure 3.2) to extend the domain of the function to variable-length inputs. The basic motivation behind using the cascade construction for domain extension was provided by the results of Merkle and Damgård [22, 54], who showed that the cascade construction applied to a suffix-free encoding[1] of the input is *collision resistant* if the underlying compression function is collision resistant.

Thus, the main security notion that has served as a guideline for the design of cryptographic hash functions, such as SHA [32], MD5 [34] etc., has been *Collision Resistance*. Indeed, these hash functions have been used to instantiate collision resistant functions in a variety of cryptographic schemes. The applications of Collision resistant hash functions (CRHFs) range from

[1]In particular, they suggest using the *Merkle-Damgård strengthening*, which involves appending the input length to the input

signature schemes (the classic “Hash-then-Sign” paradigm), to more recent applications such as those relying on the non-black-box techniques of [2].

However, the problem with using a particular security property as the guideline for hash function design is that now the requirements from hash functions extend to a large number of different security notions. Indeed, hash functions are used as *pseudorandom functions*, for *message authentication*, as *Universal One-Way Hash Functions* (UOWHFs)[2] [60], for *Key Derivation* or even as a *Random Oracle* [8].

In spite of this large variety of applications, a large fraction of the existing literature related to design and implementation of cryptographic hash functions has concentrated on collision resistance [22, 54, 15]. Apart from this, there have also been results related to pseudorandomness [5], MACs [6, 53], target collision-resistance [11, 70] and key derivation [25].

1.1.1 Hash Functions as Random Oracles

In this thesis, we start by discussing one of the most important applications of hash functions. That is, when hash functions are used to instantiate a *random oracle*. The *random oracle methodology* was introduced by Bellare and Rogaway as a “paradigm for designing efficient protocols” [8]. In this paradigm, one designs a cryptographic protocol under the assumption that there exists a *ideal random function oracle* (RO), which can be accessed by all parties in the protocol (including the adversary). Then one provides a formal

[2]also called *target collision resistant functions*

proof of security for the protocol under this assumption. In practice, the *random oracle* is instantiated using an actual cryptographic hash function, such as one of the hash functions from the SHA family [32].

It is clear that security in the ROM does not guarantee security of the scheme when instantiated with an actual hash functions. Indeed, this was shown in several "separation" results [18, 61, 4, 19, 26] which gave instances of *un-instantiable* "artificial" cryptographic schemes that are secure in the ROM. However, none of these results gave any attacks on actual schemes that were proven secure in the ROM (such as OAEP [9] or PSS [10]). Thus, the random oracle methodology is still a useful tool for designing efficient cryptographic schemes with "reasonable security guarantee".

In chapter 3, we study the design principles for cryptographic hash functions when used to instantiate the *Random Oracle*. As we discussed, an actual hash function $H : \{0,1\}^* \rightarrow \{0,1\}^n$ is designed to work on variable length inputs. Thus, one would assume that if this hash function H is "random and unstructured" enough, then there should not be any issues with using H for instantiating the random oracle (RO). However, in reality, this thinking is erroneous.

As we noted above, practical hash functions are designed by applying a domain extension technique to a fixed-length input compression function $f : \{0,1\}^m \rightarrow \{0,1\}^n$. While most of the ad-hoc design effort goes into the compression function h, the domain extension technique used in almost all hash functions is the *plain Merkle-Damgård construction* (figure 3.2). Thus, it would be unreasonable to expect such a structured construction to behave like a monolithic random oracle. On the other hand, it is a much more difficult task to design a monolithic "unstructured" hash function from scratch.

Hence, we approach this problem from a perspective of *designing a variable length input random oracle (VIL-RO) from a fixed-length input primitive (for eg., a FIL-RO)*, so that all the design effort can then be concentrated on coming up with a construction for the fixed-length primitive (in practice, the compression function).

We start by noting that none of the previous "domain-extension" results for hash functions (collision-resistance, pseudo-randomness etc.) imply a similar domain extension result for random function oracle. The main reason being that an RO construction must replicate all the properties of the random oracle, such as *pseudorandomness*, *extractability*, *programmability* etc. Since none of the previous definitions guarantee all these properties, it is not even clear how to approach this problem.

Indifferentiability

We start by discussing what it means to implement an variable-length input random oracle H from a fixed-length building block, such as a FIL-RO f. We show that the notion of *indifferentiability* introduced by Maurer et al [52] is the right definition in this context. In particular, if we show that the construction H using a fixed-length building block f is indifferentiable from a random oracle under the assumption that f is ideal, then we can use the construction H to instantiate the random oracle in any scheme provably secure in the ROM. And the resulting scheme will be secure in the *idealized model* corresponding to the primitive f.

In order to illustrate this security notion, consider a proposed RO construction C_H^f in the f-ideal model. This is an indifferentiable RO construction if there is a simulator S^H that can simulate the role of the fixed-length primitive f in the random oracle model. That is, for any attacker $A^{(\cdot,\cdot)}$ that expects access to two oracles, the following two scenarios are indistinguishable: first, where it has oracle access to the RO H and the simulator S^H and second, where it has oracle access to the RO construction C_H^f and the fixed-length primitive oracle f. Thus, S_H should essentially simulate the role played by the fixed-length primitive f with respect to the RO construction. More details on this definition are given in chapter 2, section 2.2.

Domain Extension for Random Oracle

Equipped with a suitable definition, we attempt to find an *indifferentiable construction* of a variable-length random oracle H from a fixed-length random function oracle f [3]. We start by discussing some existing, and seemingly secure domain extension techniques under this definition.

In particular, we show that the popular hash-then-sign paradigm is not secure in this context. Moreover, even the plain Merkle-Damgård construction, used in almost all existing hash functions, is not an indifferentiable

[3]This is a fixed-length random function that is accessible to all the parties in the protocol.

construction of a VIL-RO (even with *Merkle-Damgård strengthening*). Thus, the existing design principle behind hash functions such as SHA-1 or MD5 is not secure for our goal.

Therefore, instead of giving new and practically unmotivated constructions, we come up with *minimal* changes to the plain Merkle-Damgård construction that are *easily implementable in practice*, and satisfy our security definition. In particular, we propose the following modifications to the plain MD construction:

1. *Prefix-free encoding*: we show that if the inputs to the plain MD construction are guaranteed to be *prefix-free*, then the resulting construction is secure.

2. *Dropping some output bits*: we show that by dropping a non-trivial number of output bits from the output of the plain MD construction, we get an indifferentiable construction of a VIL-RO.

3. *The NMAC construction* (see figure 3.3a): we show that by applying an independent hash function g to the output of the plain MD construction using f (as in the NMAC construction [5]), then we get an indifferentiable VIL-RO construction in the random oracle model for f and g.

4. *The HMAC construction* (see figure 3.3b): we show a slight variant of the NMAC construction allows us to build the second function g from f itself (as in [5], in going from NMAC to HMAC)! In this latter

variant, one implements a secure hash function H by making two *black-box calls to the plain MD construction* (with the same IV and a given compression function f).

Ideal Cipher to Random Oracle

In practice, most hash functions are block-cipher based, either explicitly as in [15] or implicitly as in SHA-1. Therefore, we consider the question of constructing a VIL-RO H from an *ideal block cipher* $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$. An *ideal block cipher* is an ideal primitive that takes a κ-bit key, and defines an independent random permutation for each key.

We concentrate on using the Merkle-Damgård construction with the *Davies-Meyer compression function* $f(x, y) = E_y(x) \oplus x$, since this is the most practically relevant construction. One could hope to first show that the Davies-Meyer compression function is an indifferentiable construction of a FIL-RO in the ideal cipher model for E and then use one of the secure constructions of a VIL-RO from a FIL-RO. However, as we show, this first attempt fails and the Davies-Meyer construction fails to give a FIL-RO from an ideal cipher. Fortunately, we show, via direct proofs, that all four fixes proposed for FIL-RO to VIL-RO construction also work when used with the Davies-Meyer compression function in the ideal cipher model.

1.1.2 Getting the Best out of Existing Hash Functions

Having discussed the use of hash functions for instantiating the random oracle, we then analyze security of hash functions in a more general perspective. As we mentioned above, hash functions are required to satisfy a variety of different security requirements in cryptographic schemes. In fact, in the past, hash functions were viewed by practitioners as black-boxes with magic properties.

However, this perception has changed since the recent attacks on existing hash functions, including the SHA-1 and MD5. Most notable of these were the new and improved collision-finding attacks proposed by Wang et al [72, 73]. Along with other results demonstrating weaknesses of existing hash function constructions, such as [43, 45], these attacks showed that the collision-resistance of these hash functions is much worse than what was anticipated earlier. Moreover, these results have also cast a doubt on the security of these hash functions with respect to other notions.

These results have prompted NIST into organizing a series of workshops [62] for coming up with constructions for the "next generation" hash functions, and rightly so. However, this new standard is not expected to be decided any time soon. Meanwhile, practitioners are stuck with either using existing, known to be "insecure", hash functions or using an ad-hoc implementation that has not undergone the thorough analysis that standardized hash functions go through. In either case, the resulting application will be prone to possible weaknesses that are avoidable.

In chapter 4, we address this problem by looking for fixes that would allow practitioners to use standardized hash functions while side-stepping several of the weaknesses of existing constructions. As we have discussed, almost all existing hash functions are based on the plain MD construction (with Merkle-Damgård strengthening). Thus, we look for black-box fixes that can be implemented on top of the plain MD construction for several of the applications that hash functions are often used for.

Efficient Black-Box Fixes to Existing Hash Functions

Most of the prior work for hash functions has been aimed at finding iterative techniques (usually, some variants of the plain MD construction) for extending the domain of fixed length primitive to get an arbitrary length primitive satisfying the same security property, which are also often called *property-preserving transforms.* For instance, the results from chapter 3 for constructing a VIL-RO from a FIL-RO fall under this category. However, we note that it is not always the case that these variants of the plain MD construction can be implemented on top of a plain MD based hash function. An example in this context is the PRF domain extension technique in [5]. In fact, most often the reason for such "non-black-box variants" of the plain MD construction is that no black-box variants are known that preserve the required security property.

In chapter 4, our focus will be slightly different in the sense that we will emphasize this alternative goal for domain extension techniques more than property preservation. In particular, we are willing to make slightly stronger assumptions on the fixed-length primitive in order to get a variable-length primitive with a desired security property. We will look for efficient variants of the plain MD construction that satisfy the following axioms:

1. It should consist of one or two *"black-box" calls* to plain MD construction.
2. The construction must support variable-length inputs.
3. Compared to a single evaluation of the plain MD construction, its evaluation should make at most a fixed (small constant) number of extra calls to the underlying compression function.

Such a variant of the plain MD construction will allow a practitioner, who understands the security property he/she needs from the hash functions, to use an existing standardized implementation without having to tinker with the, often rather involved, internals of the implementation. We also refer to such a variant of the plain MD construction as an *efficient black-box hash function mode of operation.*

Security Properties vs. Modes of Operation

The axioms that we require our hash function modes of operation to satisfy leave very little choice for the domain extension techniques that one can use. We discuss most of the popular hash function modes of operation that satisfy our axioms:

1. *Plain MD Construction:* This captures the notion that the application uses the hash function as it is.

2. *Encode-then-MD Construction:* In this case, the user encodes the hash function input before applying the plain MD construction. Examples of popular encoding schemes used are suffix-free encoding and prefix-free encoding.

3. *MD-then-Chop Construction:* Here the user applies the plain MD mode and only uses part of the output while discarding the remaining bits. In particular, existing hash functions SHA-224 and SHA-384 are obtained this way from SHA-256 and SHA-512, respectively.

4. *NMAC/HMAC Construction:* The version of the NMAC construction that we consider simply composes two applications of the plain MD mode with possibly different initialization vectors IV_1 and IV_2. While not obeying the first axiom, the NMAC construction serves as a nice abstraction for the HMAC construction which does satisfy all our axioms (but is slightly harder to formally analyze in some cases). Essentially, the HMAC construction simulates the two black-box calls of the NMAC construction with different IVs, by adding prefixes to the input in each call.

We analyze each of these hash function modes of operation for most of the security properties that are usually desired of hash functions. The hash function properties that we analyze include: (1) Collision-resistance, (2) Pseudorandomness, (3) Message Authentication, (4) Random Oracle, (5) Target Collision Resistance (UOWHFs), (6) Second Preimage Resistance, (7) Randomness Extraction, and (8) One-Wayness.

In each case, we find the minimal assumptions that one needs to make on the compression function in order to achieve the required security property from the resulting hash function mode of operation. In many cases, it turns out that we need to make stronger assumptions on the compression function than the desired security property. Some of these results follow directly from previous work, while for other results we provide separate proofs in chapter 4.

We provide a detailed "security property vs. hash function mode of operation guide" that gives the minimal assumptions one needs to make on the compression function for each of an efficient black-box mode of operation to satisfy each of the security property (see figure 4.1). This will serve as a useful guide for practitioners on how to use existing hash functions when they desire a certain security property from them.

1.2 Block Ciphers

In the second part of this thesis, we discuss another important cryptographic primitive, a block cipher. A block cipher $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ takes a κ-bit key, and gives a permutation on n-bit strings for each key. Examples of actual block ciphers include *Data Encryption Standard* (DES), *Advanced*

Encryption Standard (AES) etc. The initial use of block ciphers was for symmetric key encryption.

Though the uses for block ciphers are not as wide-ranging as in the case of hash functions, these primitives are also used in several scenarios other than for privacy. For instance, these are used in the popular message authentication mode, CBC-MAC, or in instantiating schemes in the ideal cipher model [15, 23, 30, 42, 46].

1.2.1 Feistel Networks and Luby-Rackoff's Result

Feistel networks form the basis of several block cipher constructions, such as DES, Triple DES, Blowfish etc. A *Feistel network* consists of multiple iterative applications of the *Feistel transform.* The Feistel transform provides a construction of a permutation on $2n$-bit strings using a length-preserving function $f : \{0,1\}^n \to \{0,1\}^n$. It is defined as follows: $\Psi_f(x) \overset{def}{=} x_R \parallel (x_L \oplus f(x_R))$. The different iterative applications of the Feistel transform are known as the *rounds of the Feistel network*, and the corresponding functions are called *round functions.*

Initially, there was no theoretical justification for the usage of the Feistel networks in the design of block ciphers. This theoretical justification was provided by the result of Luby and Rackoff [47], who showed that 4 rounds of the Feistel network with independent *pseudorandom functions* in each round gives a *(strong) pseudorandom permutation* [4]. Since the paper of Luby-

[4]A strong pseudorandom permutation is indistinguishable from a truly random permu-

Rackoff, several improvements were made to their result (see [58, 51, 69, 64]).

All these results showed essentially argued the pseudorandomness of a multiple round Feistel network with pseudorandom round functions (with improving exact security of the reductions or under slightly different attack scenarios). These results provided enough justification for the use of the Feistel network based block ciphers for symmetric key encryption. Indeed, pseudorandomness of ciphertexts is the security property that one desires from a symmetric key encryption scheme.

1.2.2 Looking Beyond Pseudorandomness?

However, there are several reasons to look for other security properties from block ciphers.

(a) As we mentioned above, block ciphers are utilized for a much wider range of applications than for symmetric key encryption alone. These applications often require security properties that may be different from pseudorandomness.

(b) The round functions (or S-Boxes in actual constructions) in Feistel network based block ciphers are designed based on heuristics, and may not be (possibly even close to) pseudorandom functions. In this case, all of the previous results for the Feistel network become inapplicable.

(c) Moreover, the round functions in actual constructions may leak a lot of

tation for any attacker that can make both forward or inverse permutation queries information about the intermediate round values of the Feistel network. Again, all of the prior results for Feistel networks assume the secrecy of all (or at least some) of the round values.

In part II of this thesis, we analyze the Feistel networks from this perspective. In particular, we analyze the Feistel network under both weaker as well as stronger security notions than pseudorandomness.

Firstly, we analyze the situation when the round functions of a Feistel network are not pseudorandom functions. In particular, we analyze the situation when the round functions satisfy some weaker security property than pseudorandomness, or if the intermediate round values of the Feistel network are somehow (possibly thorough weakness of round functions) leaked to the attacker. We give positive results in such a situation in chapter 6.

Secondly, we ask if the Feistel network could be used to design a much stronger primitive than a pseudorandom permutation. That is, we analyze if, under some (ideal) security assumption on the round functions, the Feistel network is an indifferentiable construction of an *ideal block cipher*. Note that this is also the other direction of one of the questions addressed in chapter 3. We give a partial positive answer to this question in chapter 7.

1.2.3 An Abstraction for Feistel Networks

As we discussed, most of the previous results become inapplicable if either the round functions are not pseudorandom, or (at least some of) the round values are not hidden from the attacker. In order to handle this problem, we start out by discussing a combinatorial abstraction for the multiple round Feistel network that is applicable to scenarios where one or both of these assumptions do not hold. In particular, we do not make any assumptions on the round functions when stating this result.

Consider a k-round Feistel network that defines a permutation on $2n$ bits based on k length-preserving functions on n-bits. We will refer to the inputs to each of these round functions as the round values of the Feistel network. We study a game between this k-round Feistel network and an attacker that makes $2n$-bit forward/inverse permutation queries to this Feistel network and gets the result as well as all the intermediate round values. The attacker wins the game if it makes two queries such that the middle $((k/2)^{th})$ round values in these queries collide.

We show that if the attacker wins after making q queries to the k-round Feistel network in this game, then:

- **(a)** Either the number of queries, q, made by it is exponential in k.
- **(b)** Or a new round function output can be represented as an XOR of upto 5 other round values that already existed before this round function output. We refer to this as the *5-XOR condition* (see section 5.1).

The second condition essentially implies that for some query made by the attacker, a round function, say $f_i(R_i)$, where output can be represented as an XOR of upto 5 round values that were defined before this round function output. This includes round values from earlier queries, or round values from this query that were defined before this round function output.

This property essentially proves a property of an interaction between an attacker and the Feistel network that does not depend on the round functions used in the construction. We use this property for our problem by showing that if the round functions of the Feistel network are chosen such that the *5-XOR condition* is not satisfied for any efficient attacker, then the number of queries made by a "winning" attacker must be exponential in the number of rounds k (which is super-polynomial in the security parameter λ for $k = \omega(\log \lambda)$.

Moreover, we show that this result is tight in the sense that for a Feistel network with upto logarithmic number of rounds $k = \mathcal{O}(\log \lambda)$, there is an attacker that can find the input corresponding to any permutation output by making only forward queries. This is sufficient to see that the combinatorial property above does not hold for such a Feistel network. In fact, as we show in chapter 6, this implies that such a Feistel network is not useful for most applications where the round values are revealed to the attacker.

1.2.4 New and Improved Primitives

In chapter 6, we show new (or improved) constructions of some cryptographic primitives using the combinatorial property above. First, prove a stronger result than Luby-Rackoff (and subsequent results) for PRPs, that with a super-logarithmic number of rounds, the Feistel network, with independent

PRFs as round functions, is a (strong) pseudorandom permutation *even if the PRP attacker can observe the intermediate computations of the Feistel network*. This gives a more resilient PRP construction.

Coming back to our first question, we ask if there is a weaker property of the round functions than pseudorandomness, that guarantees some security property for the Feistel network. We show that even if the round functions of a super-logarithmic round Feistel network are only *unpredictable functions* (UFs) then it is an *unpredictable permutation* (UP) [5]. In fact, we show that this result is tight, in the sense that for upto a logarithmic number of rounds, there is a set of UFs that do not give a UP via the Feistel network (see lemma 23).

Next, we show that our result is also useful in a scenario where the application may need to explicitly reveal all the intermediate round values to an attacker. For instance, this comes up when one tries to add *verifiability* to the PRP or UP constructions above. The notion of *verifiable (pseudo)random functions* (VRFs) was introduced by Micali et al. [55]. These are essentially verifiable analogs of PRFs, with a public key PK and secret key SK. Given both the public and secret keys, one can compute the output y of the VRF on an input x, as well as construct a short proof that y is indeed the output of the VRF on input x and not some "garbage value" (which could easily be done for a normal PRF). On the other hand, given only the public key PK,

[5]Roughly speaking, an unpredictable function guarantees that no attacker can predict the output of the function on an unqueried input (similarly for unpredictable permutations with both forward/inverse queries).

one can verify this proof to learn whether y is indeed the correctly computed output (see formal defns. in section 6.1).

We introduce the notion of *verifiable (pseudo)random permutations* (VRPs) that are similar verifiable analogues of PRPs (or permutation analogues of VRFs). For VRPs, one can compute (and provide proofs for) both the forward and inverse permutation given the public and secret keys. We show that a super-logarithmic round Feistel network with independent VRFs as round functions, is a secure VRP. Note that in this case, the VRP proof will simply consist of intermediate round function input/output pairs along with the corresponding VRF proofs. Thus the round values need to be revealed to the attacker, which makes all of the previous techniques for the Feistel network inapplicable. Moreover, this also implies that super-logarithmic number of rounds are both necessary and sufficient.

Finally, we consider the case of *verifiable unpredictable permutations* (VUPs). These are verifiable analogs of unpredictable permutations. The corresponding notion of *verifiable unpredictable functions* (VUFs) was also introduced by Micali et al. [55]. These are also known as *unique signatures* (see [38, 49]). Micali et al. used VUFs as an intermediate step for constructing VRPs.

Note that in this case, if one uses the Feistel network with VUFs as round functions to construct VUPs, then neither are the round functions pseudorandom, nor are the round values hidden from the attacker (and are revealed as part of the VUP proof). However, we show that even in this case, a super-logarithmic round Feistel network is both necessary and sufficient to construct VUPs from VUFs.

Applications

We then provide various examples of natural scenarios where our technique (and the constructions we derive from it) are useful. These applications are described in section 6.3 (chapter 6).

- We show how our results provide a "closer-to-reality" justification for the number of Feistel rounds heuristically used in practical block cipher constructions.

- Using our results, we provide the most efficient domain extension technique for length-preserving MACs without introducing any new assumptions.

- We show that VRPs immediately yield *non-interactive, setup-free, perfectly-binding commitment schemes.*

- VRPs can be used to fix a subtle security flaw in the non-interactive lottery system of Micali-Rivest [56].

- We show that these primitives can also be used to implement so called "invariant signatures" needed by Goldwasser and Ostrovsky [38].

- Other applications of VRPs, such as verifiable CBC encryption/decryption, verifiable huge (pseudo)random objects [36] or a "proof-transferable" implementation of the Ideal Cipher Model using a semi-trusted third party.

1.2.5 The Ideal Cipher Model

In chapter 7, we analyze if the Feistel network can be used to achieve a stronger security notion than pseudorandomness. That is, we analyze if the Feistel network can be used to get an *indifferentiable construction of the ideal cipher (IC) from a random oracle (RO)*. This is essentially the converse of a question we studied in chapter 3. There we gave indifferentiable constructions of the random oracle from the ideal cipher oracle. If the converse result also holds, then it will also imply that the *ideal cipher model* (ICM) is equivalent to the *random oracle model* (ROM). Although, the ideal cipher model has not been as widely applicable as the random oracle model, there have been some results that utilize this model (see [15, 23, 30, 42, 46]).

We give a "partial positive" answer to this question, by showing that with sufficient number of rounds a Feistel network based construction using RO is *indifferentiable from the ideal cipher in the "honest-but-curious" model.* This is a weaker security notion than general indifferentiability, that is still stronger than classical indistinguishability (that is used in the case of PRPs).

Indifferentiability in the Honest-but-Curious Model

We start out by introducing the notion of indifferentiability in the honest-but-curious model in section 7.1. In order to illustrate this security notion, consider a construction $C_{\mathcal{E}}^{\mathcal{H}}$ of the IC $\mathcal{E}$ using the RO $\mathcal{H}$. Under the general notion of indifferentiability this construction is a secure ideal cipher construction, if there is an efficient simulator that can simulate the role of the RO $\mathcal{H}$ in the ideal cipher model. In this weaker notion, the task of the simulator is simply to simulate the interaction between the RO $\mathcal{H}$ and the construction $C_{\mathcal{E}}^{\mathcal{H}}$ in the ideal cipher model. That is, for any attacker A that has expects access to the ideal cipher construction oracle $C_{\mathcal{E}}^{\mathcal{H}}$ and can make queries to this construction where it observes the queries that $C_{\mathcal{E}}^{\mathcal{H}}$, in turn, makes to the random oracle $\mathcal{H}$, the following two scenarios are indistinguishable: first, where it has oracle access to $C_{\mathcal{E}}^{\mathcal{H}}$ and can observe the actual interaction between $C_{\mathcal{E}}$ and $\mathcal{H}$ or second, where it has oracle access to the ideal cipher $\mathcal{E}$ and the simulator S generates a fake interaction for the attacker.

We show that if an ideal cipher construction is indifferentiable in the honest-but-curious model, then any cryptographic protocol that is secure against honest-but-curious attackers in the ideal cipher model can also be instantiated in the random oracle model using this construction.

Next, we define the notion of a *transparent construction*, which are constructions for which general indifferentiability is equivalent to indifferentiability in the honest-but-curious model. Roughly speaking, for a transparent ideal cipher construction using RO, an attacker can query the RO indirectly by making queries to the construction and observing its interaction with the RO.

An HBC Indifferentiable construction and going beyond...

Next, we analyze the Feistel network to find out if it can give us an indifferentiable IC construction using RO. We first show that for upto a logarithmic number of rounds $k = \mathcal{O}(\log \lambda)$, the k-round Feistel network is a *transparent construction*. That is, if one can prove the honest-but-curious indifferentiability of this construction, then it will also imply general indifferentiability. This implies that if such a k-round Feistel network is an HBC indifferentiable ideal cipher construction, then the random oracle model and the ideal cipher model are equivalent! We conjecture that this is the case and that in fact, even a 6-round Feistel network might be an indifferentiable ideal cipher construction. However, we have not been able to come up with a formal proof of this conjecture.

However, we show that with super-logarithmic number of rounds $k = \omega(\log \lambda)$, the k-round Feistel network is HBC indifferentiable from the ideal cipher. This result uses the combinatorial property that we prove in chapter 5. This would indicate that one might be able to show that such a construction is indifferentiable from the ideal cipher in general, by showing that this is a transparent construction. Unfortunately, we prove that this cannot be the case by showing that for super-logarithmic number of rounds, the Feistel network cannot be a transparent construction. Thus, in this case, honest-but-curious indifferentiability is a strictly weaker notion than general indifferentiability.

Finally, we state a result of Coron [20] who shows that for upto 5 rounds, the Feistel network does not even give an HBC indifferentiable ideal cipher construction. We give a proof of this fact for a 4-round Feistel network in section 7.2.4. This result also implies that the notion of indifferentiability in the honest-but-curious model is strictly stronger than classical indistinguishability, since 4 rounds are sufficient in the latter case [47].

Chapter 2

Preliminaries

2.1 Pseudorandomness and Indistinguishability

Let $\lambda \in \mathbb{N}$ denote the security parameter. Let $\{A_\lambda, B_\lambda\}_{\lambda \in \mathbb{N}}$ be a sequence of pairs of sets. For the purposes of this thesis, A_λ and B_λ will be of the form $\{0,1\}^{n(\lambda)}$ and $\{0,1\}^{m(\lambda)}$, respectively. Here $n(\cdot)$ and $m(\cdot)$ are polynomial functions $\mathbb{N} \mapsto \mathbb{N}$. When no ambiguity can arise, we will simply represent these sets as $\{0,1\}^n$ and $\{0,1\}^m$.

Let F_λ be the set of all functions $A_\lambda \mapsto B_\lambda$, and let P_λ be the set of all permutations on A_λ. A *function ensemble* $H = \{H_\lambda\}_{\lambda \in \mathbb{N}}$ is a sequence such that each H_λ is distributed on F_λ. Here H is the *uniform function ensemble* if H_λ is uniformly distributed on F_λ. A *permutation ensemble* $H = \{H_\lambda\}_{\lambda \in \mathbb{N}}$ is a sequence such that each H_λ is distributed on P_λ, and H is the *uniform* *permutation ensemble* is H_λ is uniformly distributed on P_λ.

A function ensemble is efficiently computable if the distribution H_λ is efficiently samplable and the functions in H_λ can be computed efficiently. That is, there exist probabilistic polynomial time Turing machines, I and V, and a mapping from strings to functions, ϕ, such that (1) $\phi(I(1^\lambda))$ and H_λ are identically distributed and (2) $V(i,x) = (\phi(i))(x)$ so that $V(I(1^n), \cdot)$ is essentially $H_\lambda(\cdot)$. We denote by f_i the function assigned to i (i.e. $f_i \overset{def}{=} \phi(i)$). We refer to i as the key of f_i and to I as the key generating function of F.

Throughout this thesis, when we consider function (or permutation ensembles), the sequence of sets $\{A_\lambda, B_\lambda\}_{\lambda\in\mathbb{N}}$ will be of the form $\left\{\{0,1\}^{n(\lambda)}\right.$, $\left.\{0,1\}^{m(\lambda)}\right\}$, where n, m are functions on $\mathbb{N} \mapsto \mathbb{N}$. The usual *key generation function* I will simply output a uniformly sampled random bit string from a set $\{0,1\}^{k(\lambda)}$, i.e. $I(1^\lambda)$ is uniformly distributed over $\{0,1\}^{k(\lambda)}$.

We start by describing the notion of *indistinguishability* of two function (or permutation) ensembles. In this notion, the *distinguisher* is an oracle machine that is given oracle access to a function in F_λ or a permutation in P_λ. On input 1^λ, the distinguisher makes queries to the function or permutation that it has oracle access to, and outputs a single bit. We assume that on input 1^λ, the distinguisher only makes queries in A_λ. For the purpose of this thesis, the oracle machine can be thought to be an oracle Turing machine.

Let D be an oracle machine, let f be a function in F_λ and let H_λ be distributed over F_λ. We denote by $D^f(1^\lambda)$, the output distribution of D when its oracle queries are answered by f, and denote by $D^{H_\lambda}(1^\lambda)$, the output distribution of D when its oracle queries are answered by a function distributed according to H_λ. We will also consider oracle machines that take oracle access to a permutation in P_λ and its inverse. Let π be a permutation in P_λ and let H_λ be a distribution over P_λ. We denote by $D^{\pi,\pi^{-1}}(1^\lambda)$, the output distribution of D when it is given oracle access to the permutation π, and denote by $D^{H_\lambda,H_\lambda^{-1}}(1^\lambda)$, the output distribution of D when it is given oracle access to a permutation distributed according to H_λ.

Definition 1 ((t, q, ϵ)-indistinguishability). *Let $H = \{H_\lambda\}_{\lambda\in\mathbb{N}}$ and $\tilde{H} = \{\tilde{H}_\lambda\}_{\lambda\in\mathbb{N}}$ be two function ensembles. We say that H and $\tilde{H}$ are (t, q, ϵ)-*indistinguishable *function ensembles if for any probabilistic oracle machine D running in time t and making at most q oracle queries,*

$$\left|\Pr\left[D^{H_\lambda}(1^\lambda) = 1\right] - \Pr\left[D^{\tilde{H}_\lambda}(1^\lambda) = 1\right]\right| \le \epsilon$$

Here t, q and ϵ are all functions of the security parameter λ. The same definition can also be used for (t, q, ϵ)-indistinguishability of two permutation ensembles $\langle H, H^{-1}\rangle$ and $\langle \tilde{H}, \tilde{H}^{-1}\rangle$.

Definition 2 (negligible function). *A function $h : \mathbb{N} \to \mathbb{N}$ is* negligible *if for every constant $c > 0$ and all sufficiently large n,*

$$h(n) < \frac{1}{n^c}$$

We will say that two function (or permutation) ensembles are (computationally) indistinguishable if they are (t, q, ϵ)-indistinguishable with ϵ negligible for every polynomial t and q.

Next, we will use this notion of indistinguishability to introduce the notions of *pseudorandom functions and permutations*.

Definition 3 ((t, q, ϵ)-PRF). *Let $H = \{H_\lambda\}_{\lambda \in \mathbb{N}}$ be an efficiently computable function ensemble and let $R = \{R_\lambda\}_{\lambda \in \mathbb{N}}$ be the uniform function ensemble. H is a (t, q, ϵ)-*pseudorandom function ensemble *if for any probabilistic oracle distinguisher D that runs in time t and makes at most q oracle queries,*

$$\left|\Pr\left[D^{H_\lambda}(1^\lambda) = 1\right] - \Pr\left[D^{R_\lambda}(1^\lambda) = 1\right]\right| \leq \epsilon$$

Definition 4 ((t, q, ϵ)-PRP). *Let $H = \{H_\lambda\}_{\lambda \in \mathbb{N}}$ be an efficiently computable permutation ensemble and let $R = \{R_\lambda\}_{\lambda \in \mathbb{N}}$ the the uniform permutation ensemble. H is a $(t, q, \epsilon$-*pseudorandom permutation ensemble *if for any probabilistic oracle distinguisher D that runs in time t and makes at most q oracle queries,*

$$\left|\Pr\left[D^{H_\lambda}(1^\lambda) = 1\right] - \Pr\left[D^{R_\lambda}(1^\lambda) = 1\right]\right| \leq \epsilon$$

Definition 5 ((t, q, ϵ)-SPRP). *Let $H = \{H_\lambda\}_{\lambda \in \mathbb{N}}$ be an efficiently computable permutation ensemble and let $R = \{R_\lambda\}_{\lambda \in \mathbb{N}}$ the the uniform permutation ensemble. H is a $(t, q, \epsilon$-*strong pseudorandom permutation ensemble *if for any probabilistic oracle distinguisher D that runs in time t and makes* *at most q oracle queries,*

$$\left|\Pr\left[D^{H_\lambda,H_\lambda^{-1}}(1^\lambda)=1\right]-\Pr\left[D^{R_\lambda,R_\lambda^{-1}}(1^\lambda)=1\right]\right|\leq\epsilon$$

2.2 Ideal Primitives and Indifferentiability

In this section, we introduce the notion of *ideal primitives* and *indifferentiability* that will be used in parts I and II. We define an *ideal primitive* as an algorithmic entity which receives inputs from one of the parties and delivers its output immediately to the querying party. Moreover, the input/output pairs of an ideal primitive satisfy an *ideal property*, which can only be approximated in practice.

In this thesis, we will concentrate on two popular ideal primitives, *random oracles* and *ideal ciphers*. A *random oracle* (RO) [8] is an ideal primitive $H : \{0,1\}^* \to \{0,1\}^n$ which provides a random output to each new query. Identical input queries are given the same answer. An *ideal cipher* is an ideal primitive that models an ideal block cipher $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$. For such a block cipher, each key $k \in \{0,1\}^\kappa$ defines an independent random permutation $E_k = E(k,\cdot)$ on $\{0,1\}^n$. The ideal primitive provides oracle access to E and E^{-1}; that is, on query $(0,k,m)$ the primitives answers $c = E_k(m)$, and on query $(1,k,c)$ the primitive answers m such that $c = E_k(m)$.

Another notion related to ideal primitives, is that of *ideal assumption models*. In such models, one simplifies the task of constructing cryptographic protocols by assuming the existence of some publicly accessible *ideal primitive oracle.* The security of a protocol in such a model is also proven under this assumption, which does not formally imply its security in general but still provides a reasonable security guarantee (as discussed in the introduction). The most popular ideal assumption models are the *random oracle model* (ROM) and the *ideal cipher model* (ICM), where one assumes the existence of a random oracle and an ideal cipher oracle respectively.

As discussed in the introduction, the notion of *indistinguishability* does not suffice to discuss the security of constructions of one ideal primitive using another. The main reason is that in such a situation, one or more of oracles are publicly available. For such a situation, the notion of *indifferentiability* of random systems, introduced by Maurer et al in [52], turns out to be the right one. Indifferentiability is essentially an extension of indistinguishability, based on ideas from the *Universal Composability framework* [17] and the model of Pfitzmann and Waidner [66]. Instead of discussing the notion of indifferentiability in the context of random systems that provide interfaces to each other (as is done in [52]), we shall use this notion in the framework of *Interactive Turing Machines* (as in [17]).

Definition 6 (Indifferentiability). *A Turing machine C with oracle access to an ideal primitive $\mathcal{G}$ is said to be (t_D, t_S, q, ϵ)* indifferentiable *from an ideal primitive $\mathcal{F}$ if there exists a simulator S, such that for any distinguisher D*

it holds that :

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

The simulator has oracle access to $\mathcal{F}$ and runs in time at most t_S. The distinguisher runs in time at most t_D and makes at most q queries. Here t_D, t_S, q and ϵ are all functions of the security parameter λ. Similarly, $C^{\mathcal{G}}$ is said to be (computationally) indifferentiable from $\mathcal{F}$ is ϵ is a negligible function of λ (for polynomially bounded t_D and t_S).

As illustrated in figure 2.1, the role of the simulator is to simulator the ideal primitive $\mathcal{G}$ so that no distinguisher can tell whether it is interacting with C and $\mathcal{G}$, or with $\mathcal{F}$ and S; in other words, the output of S should look "consistent" with what the distinguisher can obtain from $\mathcal{F}$. Note that the simulator does not see the queries made by the distinguisher to $\mathcal{F}$; however, it can call $\mathcal{F}$ directly when needed for the simulation.

In part I, the ideal primitive $\mathcal{F}$ that we try to construct will be a *random oracle*, while $\mathcal{G}$ will be either a fixed-length input random function or an ideal block cipher. Thus the construction C will use primitive $\mathcal{G}$ to emulate the random oracle $\mathcal{F}$. On the other hand, in part II the ideal primitive $\mathcal{F}$ will be the ideal cipher E, while the $\mathcal{G}$ will be a random oracle H.

It is shown in [52] that if $C^{\mathcal{G}}$ is indifferentiable from $\mathcal{F}$, then $C^{\mathcal{G}}$ can replace $\mathcal{F}$ in any cryptosystem, and the resulting cryptosystem is at least as secure in the $\mathcal{G}$ model as in the $\mathcal{F}$ model. For instance, if a block cipher based

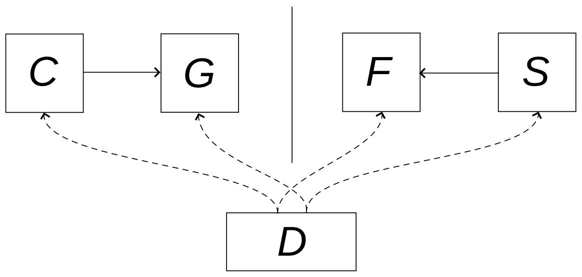


Figure 2.1: The indifferentiability notion: the distinguisher D either interacts with algorithm C and ideal primitive $\mathcal{G}$, or with ideal primitive $\mathcal{F}$ and simulator S. Algorithm C has oracle access to $\mathcal{G}$, while simulator S has oracle access to $\mathcal{F}$.

iterative hash function is indifferentiable from a random oracle H in the ideal cipher model, then the iterative hash function can replace the random oracle in any cryptosystem, and the resulting cryptosystem remains secure in the ideal cipher model if the original scheme was secure in the random oracle model.

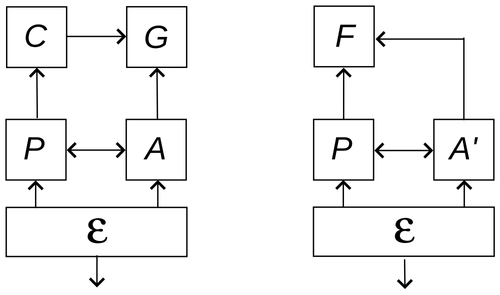


Figure 2.2: The environment $\mathcal{E}$ interacts with cryptosystem $\mathcal{P}$ and attacker $\mathcal{A}$. In the $\mathcal{G}$ model (left), $\mathcal{P}$ has oracle access to C whereas $\mathcal{A}$ has oracle access to $\mathcal{G}$. In the $\mathcal{F}$ model, both $\mathcal{P}$ and $\mathcal{A}'$ have oracle access to $\mathcal{F}$

We use the definition of [52] to specify what it means for a cryptosystem to be at least as secure in the $\mathcal{G}$ model as in the $\mathcal{F}$ model. A cryptosystem is modelled as an Interactive Turing Machine with an interface to an adversary $\mathcal{A}$ and to a public oracle. The cryptosystem is run by an *environment* $\mathcal{E}$ which provides a binary output and also runs the adversary. In the $\mathcal{G}$ model, cryptosystem $\mathcal{P}$ has oracle access to C whereas attacker $\mathcal{A}$ has oracle access to $\mathcal{G}$. In the $\mathcal{F}$ model, both $\mathcal{P}$ and $\mathcal{A}$ have oracle access to $\mathcal{F}$. The definition is illustrated in Figure 2.2.

Definition 7. *A cryptosystem is said to be at least as secure in the $\mathcal{G}$ model with algorithm C as in the $\mathcal{F}$ model, if for any environment $\mathcal{E}$ and any attacker $\mathcal{A}$ in the $\mathcal{G}$ model, there exists an attacker $\mathcal{A}'$ in the $\mathcal{F}$ model, such that*

$$\left|\Pr\left[\mathcal{E}^{\mathcal{P}^{C},\mathcal{A}^{\mathcal{G}}}(1^\lambda)=1\right]-\Pr\left[\mathcal{E}^{\mathcal{P}^{\mathcal{F}},\mathcal{A}'^{\mathcal{F}}}(1^\lambda)=1\right]\right|$$

is a negligible function of the security parameter λ. Similarly, a cryptosystem is said to be computationally at least as secure, etc., if $\mathcal{E}$, $\mathcal{A}$ and $\mathcal{A}'$ are polynomial-time in λ.

The following theorem from [52] shows that security is preserved when replacing an ideal primitive by an indifferentiable one:

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

Proof: We only provide a proof sketch; see [52] for a full proof. Let $\mathcal{P}$ be any cryptosystem, modelled as an Interactive Turing Machine. Let $\mathcal{E}$ be any environment, and $\mathcal{A}$ be any attacker in the $\mathcal{G}$ model. In the $\mathcal{G}$ model, $\mathcal{P}$ has oracle access to C whereas $\mathcal{A}$ has oracle access to ideal primitive $\mathcal{G}$; moreover environment $\mathcal{E}$ interacts with both $\mathcal{P}$ and $\mathcal{A}$. This is illustrated in Figure 2.3 (left part).

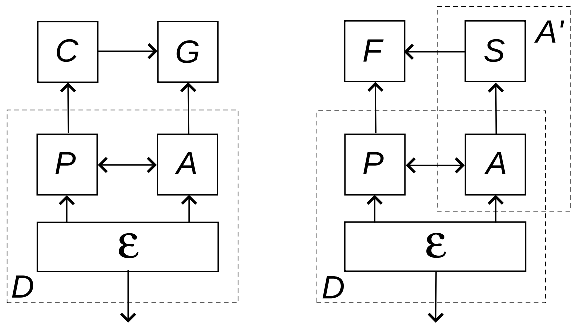


Figure 2.3: Construction of attacker $\mathcal{A}'$ from attacker $\mathcal{A}$ and simulator $\mathcal{S}$.

Since $C^{\mathcal{G}}$ is indifferentiable from $\mathcal{F}$ (see Figure 2.1), one can replace $(C, \mathcal{G})$ by $(\mathcal{F}, S)$ with only a negligible modification of the environment's output distribution. As illustrated in Figure 2.3, by merging attacker $\mathcal{A}$ and simulator $\mathcal{S}$, one obtains an attacker $\mathcal{A}'$ in the $\mathcal{F}$ model, and the difference in $\mathcal{E}$'s output distribution is negligible. $\square$

Hence if one can find an indifferentiable construction of the ideal primitive $\mathcal{F}$ using another primitive $\mathcal{G}$, then any secure cryptographic protocol in the ideal assumption model corresponding to $\mathcal{F}$ has an equivalent secure protocol in the ideal assumption model with primitive $\mathcal{G}$.

2.3 Other Cryptographic Primitives

In this section, we will give formal definitions for some of the cryptographic primitives that we will use in this thesis.

2.3.1 Message Authentication Codes

We start by defining the notion of a *Message Authentication Code*. This is a symmetric key primitive that allows a sender A to send a message m to a receiver B along with a *tag* t, such that the receiver B can verify whether the message was indeed sent by the sender A.

A Message Authentication Code, MAC, is defined over a sequence of message and tag spaces $\{\mathcal{M}_\lambda, \mathcal{T}_\lambda\}_{\lambda\in\mathbb{N}}$. It consists of a triple (Gen, Tag, Ver) of probabilistic polynomial time (PPT) algorithms:

1. The *key generating algorithm* Gen outputs the shared secret key: $s \leftarrow Gen(1^\lambda)$.

2. The tagging algorithm Tag produces a tag $t \leftarrow Tag_s(m)$ (such that $t \in \mathcal{T}_\lambda$), for any message $m \in \mathcal{M}_\lambda$.

3. The (deterministic) verification algorithm Ver produces a value $Ver_s(m,t) \in \{accept, reject\}$ indicating whether the tag t is a valid tag for message m or not.

We will define the security of MACs in the exact security framework as well.

Definition 8 ((t, q, ϵ)-secure MAC). *A Message Authentication Code, (Gen, Tag, Ver), is a* (t, q, ϵ)-secure MAC*, if for any oracle machine A that runs in time at most t, makes at most q queries to its oracles and outputs a "forgery" (m, t) such that m has never been queried to the oracle $Tag_s(\cdot)$:*

$$\Pr\left[Ver_s(m,t) = accept \,\middle|\, s \leftarrow Gen(1^\lambda);\ (m,t) \leftarrow A^{Tag_s, Ver_s}(1^\lambda)\right] \leq \epsilon$$

Here t, q and ϵ are all functions of the security parameter λ.

2.3.2 Collision Resistance

A *Collision Resistant* function ensemble is defined over a sequence of sets $\{A_\lambda, B_\lambda\}_{\lambda \in \mathbb{N}}$. It consists of a pair $(Gen, Eval)$ of PPT algorithms:

1. The *key generating algorithm Gen* outputs the function key: $s \leftarrow Gen(1^\lambda)$.

2. The function evaluation algorithm *Eval* takes a function key s and an input $x \in A_\lambda$, and maps it to an output $y \leftarrow Eval(s, x)$ such that $y \in B_\lambda$. We will also denote this as $y = h_s(x)$.

The task of an attacker in the collision resistance attack game is to find a pair of inputs for which the given function has the same output.

Definition 9 ((t, ϵ)-collision resistant function). *A efficiently computable function ensemble H_λ if a (t, e, ϵ)-collision resistant function ensemble if for* *any oracle machine A that runs in time at most t and outputs a pair of inputs $x_1, x_2 \in A_\lambda$:*

$$\Pr\left[h_s(x_1) = h_s(x_2) \,\middle|\, s \leftarrow Gen(1^\lambda);\ (x_1, x_2) \leftarrow A(1^\lambda, s)\right] \le \epsilon$$

Here t and ϵ are functions of the security parameter λ.

Part I

Hash Functions

Chapter 3

Hash Functions as Random Oracles

Random Oracle Methodology. The *random oracle model* was introduced by Bellare and Rogaway as a "paradigm for designing efficient protocols" [8]. It assumes that all parties, including the adversary, have access to a public, truly random function H. This model has proved extremely useful for designing simple, efficient and highly practical solutions for many problems. From a theoretical perspective, it is clear that a security proof in the random oracle model is only a heuristic indication of the security of the system when instantiated with a particular hash function, such as SHA-1 [32] or MD5 [34]. In fact, many recent "separation" results [18, 61, 39, 4, 19, 26] illustrated various cryptographic systems secure in the random oracle model but completely insecure for *any* concrete instantiation of the random oracle (even by a *family* of hash functions). Nevertheless, these important separation results do not seem to directly attack any of the concrete, widely used cryptosystems (such as OAEP [9] and PSS [10] as used in the PKCS #1 v2.1 standard [65]) which rely on "secure hash functions". Moreover, we hope that such particular systems are in fact *secure when instantiated with a "good" hash function.* In the random oracle model, instead of making a highly non-standard (and possibly unsubstantiated) assumption that "my system is secure with this H" (e.g., H being SHA-1), one proves that the system is at least secure with an "ideal" hash function H (under standard assumptions). Such formal proof in the random oracle model is believed to indicate that there are no structural flaws in the design of the system, and thus one can heuristically hope that no such flaws will suddenly appear with a particular, "well designed" function H. *But can we say anything about the lack of structural flaws in the design of H itself?*

BUILDING RANDOM ORACLES. From a purely theoretical view, we know that a concrete function H is not a random oracle, so to prove that a given H is "good" we need to directly argue the security of our system with this H. However, the latter task is usually unmanageable given our current tools (e.g., "realizable" properties of H such as collision-resistance, pseudorandomness or one-wayness are usually not enough to prove the security of the system). However, we argue that there is a significant gap in this reasoning. Indeed, most systems abstractly model H as a function from $\{0,1\}^*$ to $\{0,1\}^n$ (where n is proportional to the security parameter), so that H can be used on some arbitrary input domain. On the other hand, in practice such *arbitrary-length* hash functions are built by first heuristically constructing a *fixed-length* building block, such as a fixed-length compression function or a block cipher, and then iterating this building block in some manner to extend the input domain arbitrarily. For example, SHA-1, MD5, as well as all the other hash function we know of, are constructed by applying some variant of the Merkle-Damgård construction to an underlying compression function $f : \{0,1\}^{n+\kappa} \to \{0,1\}^n$ (see Figure 3.2):

```
**Function** $H(m_1, \ldots, m_\ell)$ :
    let $y_0 = 0^n$        (more generally, some fixed $IV$ value can be used)
    for $i = 1$ to $\ell$ do $y_i \leftarrow f(y_{i-1}, m_i)$
    return $y_\ell$
```

When the number of κ-bit message blocks ℓ is not fixed, one essentially appends an extra block $m_{\ell+1}$ containing the binary representation $\langle |m| \rangle$ of the length of the message (prepended by 1 and a string of 0's in order to make everything a multiple of κ; the exact details will not matter for our discussion). This procedure is known as *Merkle-Damgård strengthening.* The fixed-length compression function f can either be constructed from scratch or made out of a block-cipher E via the Davies-Meyer construction (see [74] and Figure 3.4): $f(x, y) = E_y(x) \oplus x$. For example, the SHA-1 compression function was designed specifically for hashing, but a block-cipher can nevertheless be derived from it, as illustrated in [41].

Our Main Question. Given such particular and "structured" design of our hash function H,— which is actually the design used in practice,— we argue that there exists a missing link in the claim that no structural flaws exist in the design of our system. Indeed, we only know that no such flaws exist when H was modeled as a "monolithic" random oracle, and not as an iterated hash function built from some smaller building block. As since the real implementation of H as an iterated hash function has much more structure than a random monolithic hash function would have, maybe this structure could somehow invalidate the security proof in the random oracle model? To put this into a different perspective, all the ad-hoc (and hopefully "secure") design effort for widely used hash functions, such as SHA-1 and MD5, has been placed into the design of the fixed-length building block f (or E). On the other hand, even if f (or E) were assumed to be ideal, the current proofs in the random oracle model do not guarantee the security of the resulting system when such iterated hash function H is used!

Let us illustrate our point on a well known example. A common suggestion to construct a MAC algorithm is to simply include a secret key k as part of the input of the hash function, and take for example $\mathrm{MAC}(k, m) = H(k\|m)$. It is easy to see that this construction is secure when H is modeled as a random oracle [8], as no adversary can output a MAC forgery except with negligible probability. However, this MAC scheme is completely insecure for any Merkle-Damgård construction considered so far (including Merkle-Damgård strengthening used in current hash functions such as SHA-

1, and any of the 64 block-cipher based variants of iterative hash-functions considered in [68, 15]), no matter which (ideal) compression function f (or a block cipher E) is used. Namely, given $\mathrm{MAC}(k, m) = H(k\|m)$, one can extend the message m with any single arbitrary block y and deduce $\mathrm{MAC}(k, m\|y) = H(k\|m\|y)$ without knowing the secret key k (even with Merkle-Damgård strengthening, one could still forge the MAC by more or less setting $y = \langle |m| \rangle$, where the actual block depends on the exact details of the strengthening). This (well known) example illustrates that the construction of a MAC from an iterated hash function requires a specific analysis, and cannot be derived from the security of this MAC with a monolithic hash function H. On the other hand, while the Merkle-Damgård transformation and its variants have been intensively studied for many "realizable" properties such as collision-resistance [22, 54, 68, 15], pseudorandomness [5], unforgeability [1, 53] and randomness extraction [25], it is clear that these analyses are insufficient to argue its applicability for the purposes of building a hash function which can be modeled as a random oracle, since the latter is a considerably stronger security notion (in fact unrealizable in the standard model). For a simple concrete example, the Merkle-Damgård strengthening is easily seen to preserve collision-resistance when instantiated with a collision-resistant compression function, while we just saw that it does not work to yield a random oracle or even just a variable-length MAC, and this holds even if the underlying compression function is modeled as a random oracle.

Our Goals. From the above discussion, it is clear that we need a formal definition of what it means to implement an arbitrary-length random oracle H from a fixed-length building block f or E. We have already seen that the notion of "indifferentiability" proposed in [52] is the suitable definition in this case. In particular, if we show that the construction H using fixed-length building blocks f (or E) is indifferentiable from an arbitrary length random oracle then under the assumption that f (or E) is ideal, we can use H to instantiate the random oracle in any cryptosystem proven secure in the *ROM*.

In this chapter, our goal will be to find an *indifferentiable construction* of a random oracle. However, while the notion of indifferentiability is not specific to some variant of the Merkle-Damgård transformation, we would like to give secure constructions which resemble what is done in practice as much as possible. Unfortunately, we already argued that the current design principle behind hash functions such as SHA-1 and MD5 – the (strengthened) Merkle-Damgård transformation – will *not* be secure for our ambitious goal. Therefore, instead of giving new and practically unmotivated constructions, our secondary goal is to come up with *minimal* and *easily implementable in practice* changes to the plain Merkle-Damgård construction, which would satisfy our security definition.

Our results. Based on the notion on *indifferentiability*, we provide several provably secure constructions. We start by giving three modifications to the (insecure) plain Merkle-Damgård construction which yield a secure random oracle H taking *arbitrary-length* input, from a compression function viewed as a random oracle taking *fixed-length* input. This result can be viewed as a secure *domain extender* for the random oracle, which is an interesting result of independent interest. We remark that domain extenders are well studied for such primitives as collision-resistant hash functions [22, 54], pseudorandom functions [5], MACs [1, 53] and universal one-way hash functions [11, 70]. Although the above works also showed that some variants of Merkle-Damgård yield secure domain extenders for the corresponding primitive in question, these results are not sufficient to claim a domain extender for the random oracle.

Our secure modifications to the plain Merkle-Damgård construction are the following.

1. *Prefix-Free Encoding* : we show that if the inputs to the plain MD construction are guaranteed to be *prefix-free*, then the plain MD construction is secure.

2. *Dropping Some Output Bits* : we show that by dropping a non-trivial number of output bits from the plain MD chaining, we get a secure random oracle H even if the input is not encoded in the prefix-free manner.

3. *Using NMAC construction* (see Figure 3.3a): we show that by applying an independent hash function g to the output of the plain MD chaining (as in the NMAC construction [5]), then once again we get a secure

construction of an arbitrary-length random oracle H, in the random oracle model for f and g.

4. *Using HMAC Construction* (see Figure 3.3b): we show a slightly modified variant of the NMAC construction allowing us to conveniently build the function g from the compression function f itself (as in [5] when going from NMAC to HMAC)! In this latter variant, one implements a secure hash function H by making two *black-box calls to the plain Merkle-Damgård construction* (with the same fixed IV and a given compression function f): first on $(\ell+1)$-block input $0^\kappa m_1 \ldots m_\ell$, getting an n-bit output y, and then on one-block κ-bit input y' (obtained by either truncating or padding y depending on whether or not $\kappa > n$), getting the final output.

Note that we could also define the HMAC construction by using a different initialization vector in each part of the construction, instead of using the same IV but prepending 0^κ to the input. However, our purpose here is to present these constructions as black-box extensions of existing hash functions such as SHA-1 which have only one fixed IV, in which case our proposed construction can be viewed as making two black-box calls to SHA-1 to get $SHA-1(SHA-1(0^\kappa \parallel m_1 \ldots m_\ell)$.

However, in practice most hash-function constructions are block-cipher based, either explicitly as in [68] or implicitly as for SHA-1. Therefore, we consider the question of designing an arbitrary-length random oracle H

from an ideal block cipher E, specifically concentrating on using the Merkle-Damgård construction with the Davies-Meyer compression function $f(x, y) = E_y(x) \oplus x$, since this is the most practically relevant construction. We show that all of the four fixes to the plain MD chaining which worked when f was a fixed-length random oracle, are still secure (in the ideal cipher model) when we plug in $f(x, y) = E_y(x) \oplus x$ instead. Specifically, we can either use a prefix-free encoding, or drop a non-trivial number of output bits (when possible), or apply an independent random oracle g to the output of plain MD chaining, or use the optimized HMAC construction which allows us to build this function g from the ideal cipher itself.

3.1 Domain Extension for Random Oracles

In this section, we show how to construct an iterative hash-function indifferentiable from a random oracle, from a compression function viewed as a random oracle. We start with two simple and intuitive constructions that do not work.

3.1.1 $H(x) = f(h(x))$ for Random Oracle f and Collision-Resistant One-way Hash-function h

One could hope to emulate a random oracle (with arbitrary-length input) by taking :

$$C^f(x) = f(h(x))$$

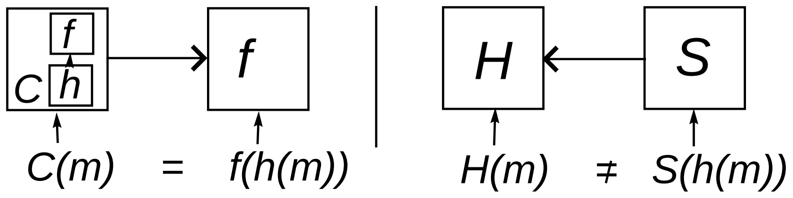


Figure 3.1: The simulator cannot output $H(m)$ since it only receives $h(m)$ and cannot recover m from $h(m)$.

where $f : \{0,1\}^n \rightarrow \{0,1\}^n$ is modelled as a random oracle and $h : \{0,1\}^* \rightarrow \{0,1\}^n$ is any collision-resistant one-way hash-function (not modelled as a random oracle). However, we show that such C^f is not indifferentiable from a random oracle; namely, we construct a distinguisher that can fool any simulator.

As illustrated in Figure 3.1, the distinguisher first generates an arbitrary m and computes $u = h(m)$. Then it queries $v = f(u)$ to random oracle f and queries $z = C^f(m)$ to C^f. It then checks that $z = v$ and outputs 1 in this case, and 0 otherwise. It is easy to see that the distinguisher always output 1 when interacting with C^f and f, but outputs 0 with overwhelming probability when interacting with H and any simulator S. Namely, when the distinguisher interacts with H and S, the simulator only receives $u = h(m)$; therefore, in order to output v such that $v = H(m)$, the simulator must either recover m from $h(m)$ (and then query $H(m)$) or guess the value of $H(m)$, which can be done with only negligible probability.

3.1.2 Plain Merkle-Damgård Construction

We show that the plain Merkle-Damgård construction (see Figure 3.2) fails to emulate a random oracle (taking arbitrary-length input) when the compression function f is viewed as a random oracle (taking fixed-length input). For simplicity, we only consider the usual Merkle-Damgård variant, although the discussion easily extends to the strengthened variant which appends the message length $\langle |m| \rangle$ at the last block :

```
**Function** $\mathrm{MD}^f(m_1, \ldots, m_\ell)$ :
    let $y_0 = 0^n$        (more generally, some fixed $IV$ value can be used)
    for $i = 1$ to $\ell$ do $y_i \leftarrow f(y_{i-1}, m_i)$
    return $y_\ell \in \{0,1\}^n$.
```

where for all i, $|m_i| = \kappa$ and $f : \{0,1\}^{n+\kappa} \to \{0,1\}^n$.

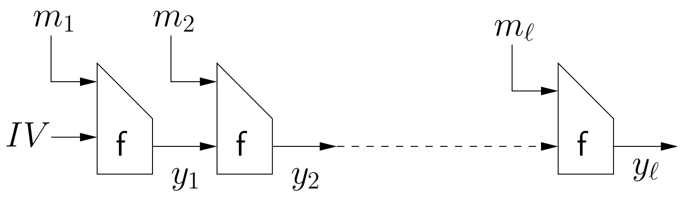


Figure 3.2: The plain Merkle-Damgård Construction

We have already mentioned in introduction a counter-example based on MAC. Namely, we showed that $\mathrm{MAC}(k, m) = H(k \| m)$ provides a secure MAC in the random oracle model for H, but is completely insecure when H is replaced by the previous Merkle-Damgård construction MD^f, because of the message extension attack. In the following, we give a more direct refutation based on the definition of indifferentiability, using again the message extension attack.

We consider only one-block messages or two-block messages. For such messages, we have that $\mathrm{MD}^f(m_1) = f(0, m_1)$ and $\mathrm{MD}^f(m_1, m_2) = f(f(0, m_1)$ $, m_2)$. We build a distinguisher that can fool any simulator as follows. The distinguisher first makes a MD^f-query for m_1 and receives $u = \mathrm{MD}^f(m_1)$. Then it makes a query for $v = f(u, m_2)$ to random oracle f. The distinguisher then makes a MD^f-query for (m_1, m_2) and eventually checks that $v = \mathrm{MD}^f(m_1, m_2)$; in this case it outputs 1, and 0 otherwise. It is easy to see that the distinguisher always outputs 1 when interacting with MD^f and f. However, when the distinguisher interacts with H and S (who must simulate f), we observe that S has no information about m_1 (because S does not see the distinguisher's H-queries). Therefore, the simulator cannot answer v such that $v = H(m_1, m_2)$, except with negligible probability.

3.1.3 Prefix-free Merkle-Damgård

In this section, we show that if the inputs to the plain MD construction are guaranteed to be prefix-free, then the plain MD construction is secure. Namely, prefix-free encoding enables to eliminate the message expansion attack described previously. This "fix" is similar to the fix for the CBC-MAC [7], which is also insecure in its plain form. Thus, the plain MD construction can be safely used for any application of the random oracle H where the length of the inputs is fixed or where one uses domain separation (e.g., prepending $0, 1, \dots$ to differentiate between inputs from different domains). For other applications, one must specifically ensure that prefix-freeness is satisfied.

A prefix-free code over the alphabet $\{0,1\}^\kappa$ is an efficiently computable injective function $g : \{0,1\}^* \to (\{0,1\}^\kappa)^*$ such that for all $x \neq y$, $g(x)$ is not a prefix of $g(y)$. Moreover, it must be easy to recover x given only $g(x)$. We provide two examples of prefix-free encodings. The first one consists in prepending the message size in bits as the first block. The last block is then padded with the bit one followed by zeroes.

```
**Function** $g_1(m)$ :
    let $N$ be the message length of $m$ in bits.
    write $m$ as $(m_1, \dots, m_\ell)$ where for all $i$, $|m_i| = \kappa$
        and with the last block $m_\ell$ padded with $10^r$.
    let $g_1(m) = (\langle N \rangle, m_1, \dots, m_\ell)$ where $\langle N \rangle$ is a $\kappa$-bit binary encoding of $N$.
```

An important drawback of this encoding is that the message length must be known in advance; this can be a problem for streaming applications in which a large message must be processed on the fly. Our second encoding g_2 does not suffer from this drawback, but requires to waste one bit per block of the message :

```
Function $g_2(m)$ :
    write $m$ as $(m_1, \ldots, m_\ell)$ where for all $i$, $|m_i| = \kappa - 1$
        and with the last block $m_\ell$ padded with $10^r$.
    let $g_2(m) = (0|m_1, \ldots, 0|m_{\ell-1}, 1|m_\ell)$.
```

Given any prefix-free encoding g, we consider the following construction of the iterative hash-function pf-$\mathrm{MD}_g^f : \{0,1\}^* \to \{0,1\}^n$, using the Merkle-Damgård hash-function $\mathrm{MD}^f : (\{0,1\}^\kappa)^* \to \{0,1\}^n$ defined previously.

```
Function pf-$\mathrm{MD}_g^f(m)$ :
    let $g(m) = (m_1, \ldots, m_\ell)$
    $y \leftarrow \mathrm{MD}^f(m_1, \ldots, m_\ell)$
    return $y$
```

Theorem 2. *The construction pf-$MD_g^f(m)$, described above, is (t_D, t_S, q, ϵ)-indifferentiable from a random oracle, in the fixed-length random oracle model for the compression function, for any t_D, with $t_S = \ell \cdot \mathcal{O}(q^2)$ and $\epsilon = 2^{-n} \cdot \ell^2 \cdot \mathcal{O}(q^2)$, where ℓ is the maximum number of κ-bit blocks in the prefix-free encoding of a query made by the distinguisher D.*

3.1.4 The Chop Solution

In this section, we show that by removing a fraction of the output of the plain Merkle-Damgård construction MD^f, one obtains a construction indifferentiable from a random oracle. This “fix” is similar to the method used by

Dodis et al. [25] to overcome the problem of using plain MD chaining for randomness extraction from high-entropy distributions, and to the suggestion of Lucks [48] to increase the resilience of plain MD chaining to multi-collision attacks. It is also already used in practice in the design of hash functions SHA-348 and SHA-224 [33] (both obtained by dropping some output bits from SHA-512 and SHA-256). Here we show that by dropping a non-trivial number of output bits from the plain MD chaining, one gets a secure random oracle H even if the input is not encoded in the prefix-free manner. For example, such dropping prevents the "extension" attacks we saw in the MAC application, since the attacker cannot guess the value of the dropped bits, and cannot extend the output of the MAC to a valid MAC of a longer message.

Formally, given a compression function $f : \{0,1\}^{n+\kappa} \to \{0,1\}^n$, the new construction chop-MD_s^f is defined as follows :

```
**Function** chop-$\mathrm{MD}_s^f(m)$ :
    let $m = (m_1, \ldots, m_\ell)$
    $y \leftarrow \mathrm{MD}^f(m_1, \ldots, m_\ell)$
    return the first $n - s$ bits of $y$.
```

Theorem 3. *The construction chop-$MD_s^f(m)$, described above, is (t_D, t_S, q, ϵ)-indifferentiable from a random oracle, in the fixed-length random oracle model for the compression function f, for any t_D, with $t_S = \ell \cdot \mathcal{O}(q^2)$ and $\epsilon = 2^{-s} \cdot \ell^2 \cdot \mathcal{O}(q^2)$. Here ℓ is the maximum number of κ-bit blocks in a query made by the distinguisher D.*

While really simple, the drawback of this method is that its exact security is proportional to $q^2 2^{-s}$, where s is the number of chopped bits and q is the number of oracle queries. Thus, to achieve adequate security level the value of s has to be relatively high, which means that short-output hash functions such as SHA-1 and MD5 cannot be fixed using this method. However, functions such as SHA-512 can naturally be fixed (say, by setting $s = 256$). A formal proof of theorem 3 is given in the next section.

3.1.5 The NMAC and HMAC constructions

The NMAC construction [5], which is the basis of the popular HMAC construction, applies an *independent* hash function g to the output of the plain MD chaining. It has been shown very valuable in the design of MACs [5], and recently also randomness extractors [25]. Here we show that if g is modelled as another fixed-length random oracle *independent* from the random oracle f (used for the compression function), then once again one gets a secure construction of an arbitrary-length random oracle H, even if plain MD chaining is applied without prefix-free encoding. Intuitively, applying g gives another way to hide the output of the plain MD chaining, and thus prevent the "extension" attack described earlier.

Formally, given $f : \{0,1\}^{n+\kappa} \to \{0,1\}^n$ and $g : \{0,1\}^n \to \{0,1\}^{n'}$, the function $\mathrm{NMAC}^{f,g}$ is defined as (see Figure 3.3a):

```
**Function** NMAC$^{f,g}(m)$ :
    let $m = (m_1, \dots, m_\ell)$
    $y \leftarrow MD^f(m_1, \dots, m_\ell)$
    $Y \leftarrow g(y)$
    return $Y$
```

Theorem 4. *The construction $NMAC^{f,g}$ is (t_D, t_S, q, ϵ) indifferentiable from a random oracle for any t_D, $t_S = \ell \cdot \mathcal{O}(q^2)$ and $\epsilon = 2^{-\min(n,n')}\ell^2\mathcal{O}(q^2)$, in the fixed-length random oracle model for the functions f and g (modelled as independent random oracles), where ℓ is the maximum number of κ-bit blocks in a query made by the distinguisher.*

To practically instantiate this suggestion, we would like to implement f and g from a single compression function. This problem is analogous to the problem in going from NMAC to HMAC in [5], although our solution is slightly different. One simple way for achieving this is to use domain separation: e.g., by prepending 0 for calls to f and 1 — for calls to g. However, with this modeling we are effectively using the prefix-free encoding mapping $m_1m_2\dots m_\ell$ to $0m_10m_2\dots0m_\ell10^\kappa$, which appears slightly wasteful. Additionally, this also forces us to go into the lower-level implementation details for the compression function, which we would like to avoid. Instead, our solution consists in applying two *black-box calls to the plain Merkle-Damgård construction MD^f* (with the same f and IV) : first to the input $0^\kappa m_1 \dots m_\ell$, getting an n-bit output y, and again to κ-bit y', where y' is defined from y as follows (see Figure 3.3b):

```
Function $\mathrm{HMAC}^f(m)$ :
    let $m = (m_1, \ldots, m_\ell)$
    let $m_0 = 0^\kappa$
    $y \leftarrow \mathrm{MD}^f(m_0, m_1, \ldots, m_\ell)$
    if $n < \kappa$ then $y' \leftarrow y \parallel 0^{\kappa - n}$
    else $y' \leftarrow y|_\kappa$
    $Y \leftarrow \mathrm{MD}^f(y')$
    return $Y$
```

Intuitively, we are almost using the NMAC construction with $g(y) = f(IV, y')$ (where y' is obtained from y as above), except we prepend a fixed block $m_0 = 0^\kappa$ to our message. This latter tweak is done to ensure that there are no inter-dependencies between using the same IV on y' and the first message block (which would have been under adversarial control had we not prepended m_0). Indeed, it is very unlikely that "high-entropy" y' will ever be equal to $m_0 = 0^\kappa$, so the analysis for NMAC can be easily extended for this optimization.

Theorem 5. *The construction $HMAC^f$, described above, is (t_D, t_S, q, ϵ) indifferentiable from a random oracle, in the fixed length random oracle model for the compression function f, for any t_D, $t_S = \ell \cdot \mathcal{O}(q^2)$ and $\epsilon = 2^{\min(n,\kappa)} \cdot \ell^2 \cdot \mathcal{O}(q^2)$. Here ℓ is the maximum number of κ bit blocks in a query made by the distinguisher D.*

The formal proofs for both theorems 4 and 5 are given in the next section.

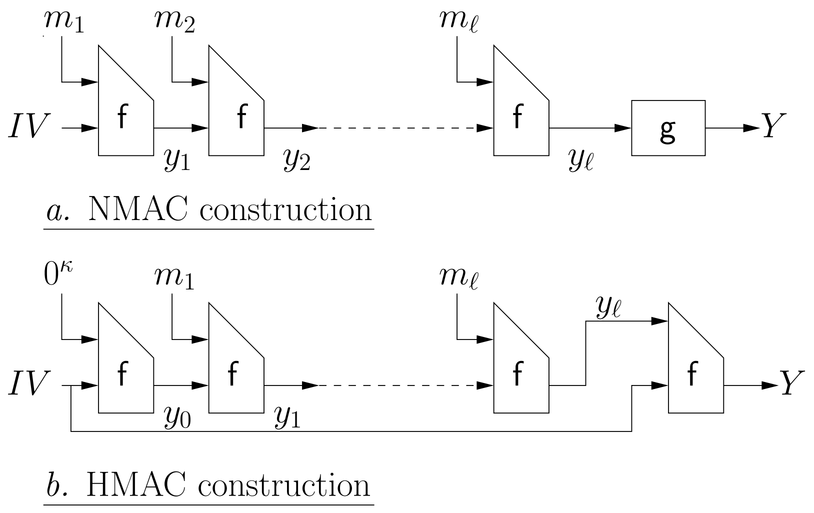


Figure 3.3: The NMAC and HMAC constructions

3.2 Constructions using Ideal Cipher

In practice, most hash-function constructions are block-cipher based, either explicitly as in [68] or implicitly as for SHA-1. Therefore, we consider the question of designing an arbitrary-length random oracle H from an ideal block cipher $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$, specifically concentrating on using the Merkle-Damgård construction with the Davies-Meyer compression function $f(x, y) = E_y(x) \oplus x$ (see Figure 3.4), since this is the most practically relevant construction. We notice that the question of designing a *collision-resistant* hash function H from an ideal block cipher was explicitly considered by Preneel, Govaerts and Vandewalle in [68], and latter formalized and extended by Black, Rogaway and Shrimpton [15]. Specifically, the authors of [15] actually considered 64 block-cipher variants of the Merkle-

Damgård transform (which included the Davies-Meyer variant among them), and formally showed that exactly 20 of these variations (including the Davies-Meyer variant) are collision-resistant when the block cipher E is modeled as an ideal cipher. However, while our work will also model E as an ideal cipher, our security goal is considerably stronger than mere collision-resistance. Indeed, we already pointed out that none of the 64 variants above can withstand the "extension" attack on the MAC application, even with the Merkle-Damgård strengthening. And even when restricting to a fixed number of blocks ℓ (which invalidates the "extension" attack), collision-resistance is completely insufficient for our purposes. For example, the authors of [15] show the collision-resistance when using the plain MD chaining with fixed IV and compression function $f(x, y) = E_y(x)$. On the other hand, it is easy to see that this method does not provide a secure random oracle H according to our definition.

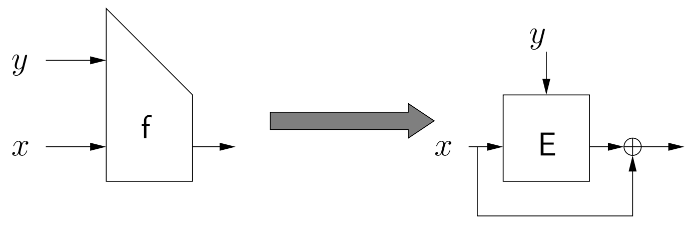


Figure 3.4: The Davies-Meyer Compression function

From a different direction, if we could show that the Davies-Meyer compression function $f(x, y) = E_y(x) \oplus x$ is a secure random oracle when E is an ideal block-cipher, then we could directly apply any of the three fixes discussed above. Unfortunately, this is again not the case: intuitively, the above construction allows anybody to compute x from $f(x, y) \oplus x$ and y (since $x = E_y^{-1}(f(x, y) \oplus x)$), which should not be the case if f was a true random oracle. Thus, we need a direct proof to argue the security of the Davies-Meyer construction. Luckily, using such direct proofs we indeed argue that all of the fixes to the plain MD chaining which worked when f was a fixed-length random oracle, are still secure when $f(x, y) = E_y(x) \oplus x$ is used instead. Namely, we can either use a prefix-free encoding, or drop a non-trivial number of output bits, or apply an independent random oracle g to the output of plain MD chaining. With respect to this latter fix, we also show that we can implement this independent g using the ideal cipher itself, similarly to the case with an ideal compression function f.

Formally, given a block-cipher $E : \{0, 1\}^\kappa \times \{0, 1\}^n \to \{0, 1\}^n$, the plain Merkle-Damgård hash-function with Davies-Meyer's compression function is defined as :

```
**Function** $\mathrm{MD}^E(m_1, \ldots, m_\ell)$ :
    let $y_0 = 0^n$        (more generally, some fixed $IV$ value can be used)
    for $i = 1$ to $\ell$ do $y_i \leftarrow E_{m_i}(y_{i-1}) \oplus y_{i-1}$
    return $y_\ell \in \{0, 1\}^n$.
```

where for all i, $|m_i| = \kappa$. The block-cipher based iterative hash-functions pf-MD_g^E, chop-MD_s^E, NMAC_g^E and HMAC^E are then defined as in section 3.1, using MD^E instead of MD^f.

Theorem 6. *The block-cipher based constructions pf-MD$_g^E$, chop-MD$_s^E$, NMAC$_g^E$ and HMAC^E are (t_D, t_S, q, ϵ)-indifferentiable from a random oracle, in the ideal cipher model for E, for any t_D and $t_S = \ell \cdot \mathcal{O}(q^2)$, with $\epsilon = 2^{-n} \cdot \ell^2 \cdot \mathcal{O}(q^2)$ for pf-MD$_g^E$, $\epsilon = 2^{-s} \cdot \ell^2 \cdot \mathcal{O}(q^2)$ for chop-MD$_s^E$, $\epsilon = 2^{-\min(n,n')} \cdot \ell^2 \cdot \mathcal{O}(q^2)$ for NMAC$_g^E$ and $\epsilon = 2^{-\min(\kappa,n)} \cdot \ell^2 \cdot \mathcal{O}(q^2)$ for HMAC^E. Here ℓ is the maximum message length queried by the distinguisher.*

Proof: We will prove that the Merkle-Damgård (MD) based constructions are indifferentiable constructions of a random oracle (RO), when applied to the Davies-Meyer (DM) compression function using an ideal block cipher (IC). The four constructions that we prove to be secure are:

1. **Prefix-free Merkle-Damgård construction pf-MD$_g^E$:** In this construction, we apply the Davies-Meyer Merkle-Damgård (DMMD) construction to a prefix-free encoding of the input (using the prefix-free encoding scheme g).

2. **Merkle-Damgård with chopped output chop-MD$_s^E$:** This is the plain DMMD construction applied directly to the input, with a non-trivial number, s, of the output bits chopped.

3. **NMAC construction NMAC^{E1,E2}:** This construction uses two independent ideal block ciphers $E1 : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ and $E2 : \{0,1\}^{\kappa'} \times \{0,1\}^{n'} \to \{0,1\}^{n'}$. It first applies the DMMD construction using $E1$ to the input, getting a n bit output Y. Then it applies

the Davies-Meyer compression function using $E2$ to Y to get the final output.

4. **HMAC construction HMAC^E:** This is an instantiation of the NMAC construction using the same ideal cipher for both parts, but using different initialization vectors in each part (implemented by prepending 0^κ to the input).

The proof of indifferentiability in each of these cases essentially involves two steps. First, we propose a simulator that simulates the task of the ideal cipher in the random oracle model (ROM). Secondly, we show that the view of any distinguisher in the ROM, with oracle access to the actual random oracle and the ideal cipher simulator, does not differ from its view in the ideal cipher model (ICM), with oracle access to the RO construction and the ideal cipher, by more than a negligible amount. We start by providing an intuitive idea of the basic paradigm used in each of the proofs, followed by the formal proofs for each case.

The Simulator. The task of the simulator in each of the cases is to simulate the ideal cipher in the ROM, in such a way that its relation with the random oracle is consistent with the relation between that actual ideal cipher and the RO construction in the ICM. Thus, in each case, the simulator essentially gives random responses to all forward block cipher queries except those that form the last application of the ideal cipher for some random oracle input (when processed using the RO construction). For example, in the Chop construction this will be the last block cipher call in the Davies-Meyer Merkle-Damgård computation.

If the query corresponds to a last block cipher call, then the simulator consults the random oracle and adjusts its response so as to remain consistent with the ICM scenario.

In the case of an inverse block cipher query, the simulator always gives random responses. In addition, the simulator also maintains a table $\mathcal{T}$ in which it records all previous query-response pairs (so as to maintain consistency among its responses).

Proof of Indifferentiability. Each of the proofs of indifferentiability consist of a hybrid argument that presents a sequence of mutually indistinguishable games starting in the random oracle model, with the RO F and the ideal cipher simulator S, leading up to the ideal cipher model, with the RO construction (which we call C^E) and the ideal cipher E. The overall structure of the hybrid argument is similar for each of the constructions, though the formal proof differs. We will describe the overall structure of the proof here.

Game 1. This is the random oracle model, where the distinguisher is given oracle access to the random oracle F and the ideal cipher simulator S.

Game 2. In this game, we introduce a *relay algorithm* R_0 that is simply a dummy algorithm between the distinguisher and the random oracle F. This relay algorithm simply relays the queries of the distinguisher to the RO and relays back the output of F.

Game 3. In this game, we modify the simulator by defining a few failure conditions for its query-response pairs. If any of these failure conditions is true, then the new simulator S_0 explicitly fails. These failure conditions capture certain collision conditions which, if they happen, could be exploited by the distinguisher to decide the scenario it is in. The failure conditions are different for each constructions and are described in the formal proof. Thus the distinguisher has oracle access to the new simulator S_0^F and the relay algorithm R_0^F in this game.

Game 4. Now we modify the relay algorithm so as to make its responses directly dependent on the simulator, instead of the RO F. The new relay algorithm R_1 essentially evaluates the construction C^E using the simulator S_0 instead of the ideal cipher E. The main idea here is to prove that unless one of the failure conditions described in game 3 is true for the query-response pairs of the simulator S_0 (in which case it would fail), the responses of R_1 are still consistent with the random oracle. Thus, games 3 and 4 form the heart of the proof in each case. In this game, the distinguisher has oracle access to the relay algorithm $R_1^{S_0^F}$ and the simulator S_0^F.

Game 5. In this game, we modify the simulator so that it chooses its responses independent of the random oracle (i.e. uniformly random by itself). In addition, the new simulator S_1 does not check for any of the failure conditions described above. This does not introduce any changes in the view of the distinguisher since the relay algorithm R_1 uses the simulator S_1 to construct its responses (which still look random). Thus, in this game the distinguisher has oracle access to the relay algorithm $R_1^{S_1}$ and the simulator S_1.

Game 6. Finally, we replace the simulator S_1 by the ideal block cipher E. Thus the relay algorithm R_1 now becomes identical to the RO construction C^E. Thus in this game the distinguisher has oracle access to the RO construction C^E and the ideal cipher E.

Now that we have the overall structure of the indifferentiability proofs, we will give the formal proofs for each of the four RO constructions. The proof of this theorem is a consequence of lemmas 1, 2, 3 and 4. $\square$

3.2.1 Prefix-free Merkle-Damgård Construction

In this section, we will give the proof of indifferentiability for the prefix-free Merkle-Damgård construction pf-MD$_g^E$.

Lemma 1. *The prefix-free Merkle-Damgård construction pf-MD$_g^E$ using an* *ideal cipher $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ is (t_D, t_S, q, ϵ)-indifferentiable from a random oracle in the ideal cipher model for E, for any t_D and $t_S = \mathcal{O}(q \cdot R_g(q \cdot \kappa))$ (where $R_g(q \cdot \kappa)$ is the running time of the decoding algorithm of g on an input of length $q \cdot \kappa$), with $\epsilon = 2^{-n} \cdot \ell^2 \cdot \mathcal{O}(q^2)$.*

Proof:

The Simulator. The simulator S_E accepts either forward ideal cipher queries, $(+, x, y)$, or inverse ideal cipher queries, $(-, x, z)$, such that $x \in \{0,1\}^\kappa$ and $y, z \in \{0,1\}^n$. In either case, the simulator S responds with a n-bit string that is interpreted as $E_x(y)$ in case of a forward query $(+, x, y)$ and as $E_x^{-1}(z)$ in case of an inverse query. The simulator maintains a table $\mathcal{T}$ of triples $(x, y, z) \in \{0,1\}^\kappa \times \{0,1\}^n \times \{0,1\}^n$, such that it either responded with z to a forward query $(+, x, y)$ or with y to an inverse query $(-, x, z)$.

On getting a forward query $(+, x, y)$, the simulator searches its table $\mathcal{T}$ for a triple (x, y, z) for any z. If there exists such a triple, then it responds with z otherwise it needs to choose a new response to this query. It then searches its table $\mathcal{T}$ for a sequence of triples $(x_1, y_1, z_1) \dots (x_i, y_i, z_i)$ such that:

- The bit string $x_1 \parallel \dots \parallel x_i \parallel x$ decodes to a valid RO input under the prefix-free encoding g.
- It is the case that $y_1 = IV$, where IV denotes the initialization vector used in the construction pf-MD$_g^E$.
- For each $j = 2 \dots i$, it is the case that $y_j = z_{j-1} \oplus y_{j-1}$.

- It is the case that $y = z_i \oplus y_i$, where y is the input message in the current forward query.

Note that for an empty sequence of triples, i.e. when just considering the κ-bit block x from the current query, only the first requirement makes sense. We additionally also require that $y = IV$ in this case.

If the simulator S finds such a sequence of triples, then it needs to give a response that is consistent with the random oracle output on $g^{-1}(x_1 \parallel \ldots \parallel x_i \parallel x)$. Thus, the simulator makes this RO query to get the output $Y = F(g^{-1}(x_1 \parallel \ldots \parallel x_i \parallel x))$, and responds with $z = Y \oplus y$. If the simulator does not find such a sequence of triples, it outputs a random response z. In either case, it stores the triple (x, y, z) in its table $\mathcal{T}$.

On receiving an inverse query $(-, x, z)$, the simulator S searches its table $\mathcal{T}$ for a triple (x, y, z) for any y. If it finds such a triple, then it outputs y as its response. If it does not find such a triple, it chooses a random n-bit string y and responds with y. It then stores the triple (x, y, z) into its table $\mathcal{T}$.

Proof of Indifferentiability. We need to prove that the distinguisher cannot tell apart the two scenarios, one where it has oracle access to the random oracle F and the simulator S and the other where it has access to the RO construction pf-MD$_g^E$ and the ideal block cipher E. As we mentioned above, the proof involves a hybrid argument starting in the random oracle scenario, and ending in the ideal cipher scenario through a sequence of mutually indistinguishable hybrid games.

Game 1. This is the *random oracle model*, where the distinguisher D has oracle access to the random oracle F and the simulator S described above. Let G_1 denote the event that D outputs 1 after interacting with F and S. Thus,

$$\Pr[G_1] = \Pr\left[D^{F,S^F}(1^\lambda) = 1\right]$$

Game 2. In this game, we give the distinguisher oracle access to a dummy *relay algorithm* R_0 instead of direct oracle access to the random oracle F. This relay algorithm R_0 is given oracle access to the random oracle F, and on getting a random oracle query from the distinguisher, it simply makes the same query to the RO F and forwards the RO output to the distinguisher as its response. Let G_2 denote the event that the distinguisher outputs 1 in this game. Since we have left the view of the distinguisher unchanged in this game, the distribution of its outputs also remains the same.

$$\Pr[G_2] = \Pr\left[D^{R_0^F,S^F}(1^\lambda) = 1\right] = \Pr[G_1]$$

Game 3. In this game, we modify the simulator S. In particular, we restrict the responses of the simulator such that they never satisfy certain specific failure conditions. If the simulator comes up with a response that results in its responses satisfying one of these conditions, then it fails explicitly instead of sending this response.

The failure conditions that the new simulator S_0 avoids essentially describe certain dependencies that could arise among its responses that could be exploited by the distinguisher. In response to a forward query $(+, x, y)$, the new simulator chooses a response $z \in \{0,1\}^n$ similar to the original simulator S and it checks for the following conditions:

1. *Condition B_1:* It is the case that $z \oplus y = IV$, where IV is the n-bit initialization vector used in the RO construction pf-MD_g^E.

2. *Condition B_2:* There is a triple $(x', y', z') \in \mathcal{T}$, with $(x', y') \neq (x, y)$, such that $y' \oplus z' = y \oplus z$.

3. *Condition B_3:* There is a triple $(x', y', z') \in \mathcal{T}$, with $(x', y') \neq (x, y)$, such that $y \oplus z = y'$.

If the response z is chosen by the simulator S_0 at random then the simulator S_0 checks for these conditions and explicitly fails if any of them holds. However, if the simulator is forced to choose a response in order to maintain consistency with the random oracle F, then it only checks for the conditions B_1 and B_2.

Let us briefly describe how the distinguisher can exploit each of these conditions to its advantage. If the condition B_1 holds then the distinguisher could possibly force two different RO query sequences to end in the same block, where one input is the suffix of the other. Hence the simulator can be consistent with at most one of these two RO inputs. If condition B_2 holds, then the distinguisher can again force two query sequences to end in the same block. However, in this case the two RO inputs have a common suffix and the simulator can be consistent with at most one of these inputs. If condition B_3 holds, then distinguisher can make a RO query sequence to the simulator such that the simulator is not consistent with the RO output because the query corresponding to the last block of the (encoding of the) RO input is not the last one that it makes.

Now we will estimate the occurrence probability for each of the above failure conditions. Let the number of random oracle queries made by the distinguisher be q_F, and let the number of ideal cipher queries be q_E. To start with, it is easy to see that the occurrence probability of condition B_1 is at most the probability that one of $q(= q_E + q_F)$ random n-bit strings are equal to IV.

To bound the occurrence probability of failure condition B_2, we will analyze three situations separately.

- Query $(+, x, y)$ does not correspond to the last block of (the prefix-free encoding of) a random oracle query. In this case, condition B_2 occurs only if the uniformly random n-bit string $y \oplus z$ (with z chosen by the simulator), collides with one of q_E n-bit strings corresponding to other queries.

- Both (x, y, z) and (x', y', z') form last blocks of random oracle queries. In this case, condition B_2 is exactly the event that two random oracle outputs collide.

- The triple (x, y, z) forms the last block of a random oracle query, but (x', y', z') does not. In this case, $y' \oplus z'$ is a random n-bit string chosen by the simulator. Hence, condition B_2 corresponds to the random oracle output $y \oplus z$ collides with a random n-bit string chosen by the simulator.

Hence, we can bound the occurrence probability of condition B_2 by the birthday bound over $(q_E + q_F)$ uniformly random n-bit strings.

The simulator checks for condition B_3 only if it chooses the response independently. In this case, the occurrence probability of this failure condition can be bounded by the $(q_E 2/2^n)$. We do not force the simulator to check for condition B_3 when it is forced to be consistent with the random oracle. This is because the distinguisher can force this condition using RO queries, but this does not help since we use a prefix-free encoding before applying the Merkle-Damgård construction.

If an inverse query $(-, x, z)$ is made to the simulator S_0, the it chooses a response $y \in \{0,1\}^n$ to this query similar to the original simulator S and checks for the following failure conditions:

1. *Condition C_1.* It is the case that $y = IV$ or $y \oplus z = IV$, where IV is the n-bit initialization vector.

2. *Condition C_2.* There is a triple $(x', y', z') \in \mathcal{T}$, with $(x', z') \neq (x, z)$, such that $y' \oplus z' = y \oplus z$.

3. *Condition C_3.* There is a triple $(x', y', z') \in \mathcal{T}$, with $(x', z') \neq (x, z)$, such that $y \oplus z = y'$ or $y' \oplus z' = y$.

In the case of inverse queries, the simulator always independently chooses random responses to any new queries and fails if any of the conditions C_1, C_2 or C_3 holds, and hence estimating the occurrence probability of these failure conditions is straightforward. The reasons for avoiding the conditions C_1, C_2 and C_3 are similar to those given above for B_1, B_2 and B_3.

Let G_3 denote the event that the distinguisher outputs 1 in game 3, i.e. $\Pr[G_3] = \Pr\left[D^{R_0^F, S_0^F}(1^\lambda) = 1\right]$. The responses of the distinguisher in games 2 and 3 differ only in situations where the new simulator S_0 explicitly fails and the original simulator S does not. This event is identical with the event that any of the failure conditions hold for the responses of either simulator (both of which are identically distributed).

$$\begin{aligned}
|\Pr[G_3] - \Pr[G_2]| &\leq 2 \cdot \Pr[B_1 \cup B_2 \cup B_3 \cup C_1 \cup C_2 \cup C_3 \text{ hold for a} \\
&\qquad \text{corresponding query.}] \\
&\leq \frac{2 \cdot (q_{_E} + q_{_F}) \cdot (2 \cdot (q_{_E} + q_{_F}) + 1)}{2^n} \\
&= \mathcal{O}\left(\frac{q^2}{2^n}\right)
\end{aligned}$$

GAME 4. In this game, we modify the relay algorithm and leave the ideal cipher simulator unchanged. The underlying idea is to make the responses of the relay algorithm directly dependent on the simulator. Thus, instead of giving the new relay algorithm R_1 an oracle access to the random oracle F, here it is given oracle access to the simulator S_0.

On a random oracle query X, the relay algorithm R_1 computes the prefix-free encoding of X, i.e. $g(X)$. It then applies the Davies-Meyer Merkle-Damgård construction to $g(X)$ by querying the simulator S_0. Thus the relay algorithm R_1 is essentially the same as the random oracle construction pf-MD_g^E, except that it is based on the simulator S_0 instead of the ideal cipher E.

Let G_4 denote the event that the distinguisher D outputs 1 when given oracle access to S_0 and R_1 in this game. Thus, we know that

$$\Pr[G_4] = \Pr[D^{R_1^{S_0}, S_0^F}(1^\lambda) = 1]$$

Now we will show that the view of the distinguisher D remains unchanged (upto a negligible additive factor) in the transformation from game 3 to game 4. We will assume that that maximum length of the prefix-free encoding $g(X)$ of a random oracle input X queried upon by the distinguisher is $\ell\kappa$. This claim is formally stated below:

Claim 7. *Let G_3 and G_4 denote the events that the distinguisher D outputs 1 in games 3 and 4, respectively. If q_E and q_F denote the number of ideal* *cipher and random oracle queries made by the distinguisher (respectively), then it is the case that*

$$|Pr[G_4] - Pr[G_3]| = \mathcal{O}\left(\frac{(q_{_E} + \ell \cdot q_{_F})^2}{2^n}\right)$$

proof of claim 7: From the view of the distinguisher, the games 3 and 4 differ only if it detects any difference in the responses of the relay algorithm or the simulator in these two games. We will prove that such a difference in the responses is impossible unless the simulator S_0 fails in either game 3 or 4. We start by demonstrating a few useful properties of the responses of the simulator S_0.

Claim 8. *If the simulator S_0 does not explicitly fail, then there are no two different sequences of κ-bit blocks $x_1 \ldots x_m$ and $x'_1 \ldots x'_p$ with corresponding triples $(x_1, y_1, z_1) \ldots (x_m, y_m, z_m)$ and $(x'_1, y'_1, z'_1) \ldots (x'_p, y'_p, z'_p)$ in table $\mathcal{T}$ such that:*

- *Both $x_1 \parallel \ldots \parallel x_m$ and $x'_1 \parallel \ldots \parallel x'_p$ constitute valid prefix-free encodings of random oracle inputs.*
- *It is the case that $y_1 = y'_1 = IV$, and for each $s = 1 \ldots m$ and $s' = 1 \ldots p$, $y_s = y_{s-1} \oplus z_{s-1}$ and $y'_{s'} = y'_{s'-1} \oplus z'_{s'-1}$.*
- *There is a $s \in \{1, m\}$ such that $(x_s, y_s, z_s) = (x'_p, y'_p, z'_p)$.*

proof of claim 8: We will prove this claim by performing an induction on the number of queries made to the simulator S_0, and show that unless the simulator explicitly fails, such sequence of triples cannot exist in the table $\mathcal{T}$ maintained by it. When no queries have been made, then the claim is vacuously true. Assume that it holds when q queries have already been made to the simulator S_0.

Say there are two sequences of κ-bit blocks $x_1 \ldots x_m$ and $x'_1 \ldots x'_p$ that satisfy the properties mentioned in the statement of the claim after the $(q+1)^{th}$ query. We can deduce that there are two subsequences of κ-bit blocks $x_{j-r} \ldots x_j$ and $x'_{p-r} \ldots x'_p$ such that:

$$\forall s \in \{0, r\} \ : \ (x_{j-s}, y_{j-s}, z_{j-s}) = (x'_{p-s}, y'_{p-s}, z'_{p-s})$$

If $r < j-1$ and $r < p-1$, then consider the triples $(x_{j-r-1}, y_{j-r-1}, z_{j-r-1})$ and $(x'_{p-r-1}, y'_{p-r-1}, z'_{p-r-1})$. Since $y_{j-r} = y'_{p-r}$, we can deduce that $y_{j-r-1} \oplus z_{j-r-1}) = y'_{p-r-1} \oplus z'_{p-r-1}$. Without loss of generality, assume that the query corresponding to the triple $(x_{j-r-1}, y_{j-r-1}, z_{j-r-1})$ was made after the one corresponding to $(x'_{p-r-1}, y'_{p-r-1}, z'_{p-r-1})$. If this query was a forward query then the simulator S_0 would have explicitly failed because of failure condition B_2. If this was an inverse query then the simulator would have failed because of failure condition C_2.

Now consider the case that $r = p-1$ but $r < j-1$. In this case, if the triple $(x_{j-r-1}, y_{j-r-1}, z_{j-r-1})$ was generated as a result of a forward query, then the simulator S_0 would have explicitly failed because of failure condition B_1 since $z_{j-r-1} \oplus y_{j-r-1} = y_{j-r} = y'_1 = IV$. If this triple was generated due to an inverse query then the simulator will fail because of failure condition C_1. The case when $r = j - 1$, but $r < p - 1$ is similar.

Lastly, if $r = p - 1 = j - 1$ then we have that $\forall s \in \{1, p\} \; : \; (x_s, y_s, z_s) = (x'_s, y'_s, z'_s)$. But this implies that $x'_1 \parallel \ldots \parallel x'_p$ is a prefix of $x_1 \parallel \ldots \parallel x_m$, which is not possible since they are encodings of two different inputs using the prefix-free encoding g.

Hence, we can conclude that there can be no such sequence of κ-bit blocks $x_1 \parallel \ldots \parallel x_m$ and $x'_1 \parallel \ldots \parallel x'_p$ if the simulator S_0 does not explicitly fail. □

Next we show that if the distinguisher wishes to find the random oracle output for an input $X \in \{0,1\}^*$, such that $g(X) = x_1 \parallel \ldots \parallel x_s$, by making queries to the simulator S_0 to compute the Davies-Meyer Merkle-Damgård construction applied to $x_1 \parallel \ldots \parallel x_s$, then the only way it can do so is by making the ordered sequence of forward queries $(+, x_1, y_1) \ldots (+, x_s, y_s)$.

Claim 9. *Consider any sequence of κ-bit blocks $x_1 \ldots x_s$, with corresponding triples $(x_1, y_1, z_1) \ldots (x_s, y_s, z_s)$ in the table $\mathcal{T}$ maintained by the simulator S_0, such that:*

- *$x_1 \parallel \ldots \parallel x_s$ is a valid encoding of a random oracle input X under the prefix-free encoding g.*
- *$y_1 = IV$, and for all $j \in \{2, s\}$ it is the case that $y_j = y_{j-1} \oplus z_{j-1}$.*

If the simulator S_0 does not explicitly fail then it must be the case that the *triples $(x_1, y_1, z_1) \dots (x_s, y_s, z_s)$ were stored as a result of the ordered sequence of queries $(+, x_1, y_1) \dots (+, x_s, y_s)$.*

proof of claim 9: To the contrary, assume that the sequence of queries that resulted in the triples $(x_1, y_1, z_1) \dots (x_s, y_s, z_s)$ was not the sequence of forward queries given in the claim statement. We can then deduce that at least one of the following must be true regarding this sequence of queries:

1. For $j \in \{1, s-1\}$, a forward query $(+, x_j, y_j)$ was made when the triple $(x_{j+1}, y_{j+1}, z_{j+1})$ already existed in the table $\mathcal{T}$.

2. For $j \in \{2, s\}$, an inverse query was made $(-, x_j, z_j)$ when the triple $(x_{j-1}, y_{j-1}, z_{j-1})$ already existed in the table $\mathcal{T}$.

3. The triple (x_1, y_1, z_1) was generated as a result of an inverse query $(-, x_1, z_1)$.

In the first case, we know from claim 8 that the triple (x_j, y_j, z_j) cannot be the last block of the prefix-free encoding of another query if the simulator S_0 does not fail. Hence it must be the case that the response to the corresponding query was randomly chosen by the simulator itself (independent of the random oracle). But since the triple $(x_{j+1}, y_{j+1}, z_{j+1})$ already exists in table $\mathcal{T}$, the simulator will explicitly fail from condition B_3 since the equality $y_j \oplus z_j = y_{j+1}$ holds. In the second case, the simulator will explicitly fail due to failure condition C_3 since the equality $y_j = z_{j-1} \oplus y_{j-1}$ holds. In the last case the simulator fails due to failure condition C_1.

Thus the simulator S_0 explicitly fails in either of the above situations, and the only sequence of queries possible is the one mentioned in the statement of the claim. $\square$

Next, we wish to show that the responses of the relay algorithm R_0 and the simulator S_0 are always consistent in game 3. Note that in game 4, the relay algorithm R_1 responds to all queries by computing the RO construction pf-MD$_g^{S_0}$, with the ideal cipher E replaced by the simulator S_0. On the other hand, the responses of the relay algorithm R_0 could be inconsistent with the simulator S_0 (i.e. the distinguisher may get a different output to a random oracle input depending on whether it uses the construction pf-MD$_g^{S_0}$ itself, or queries the relay algorithm R_0). We show that such a situation is impossible unless the simulator S_0 fails.

Claim 10. *In game* 3*, if the simulator S_0 never fails then there is no sequence of κ-bit blocks $x_1 \dots x_j$, with corresponding triples $(x_1, y_1, z_1) \dots (x_j, y_j, z_j)$ such that:*

- *The bit string $x_1 \parallel \dots \parallel x_j$ is a valid prefix-free encoding of a random oracle input.*
- *$y_1 = IV$ and for $l = 2 \dots j$ it is the case that $y_l = y_{l-1} \oplus z_{l-1}$.*
- *To the random oracle query $g^{-1}(x_1 \parallel \dots \parallel x_j)$, the response of the relay algorithm R_0 is different from $y_j \oplus z_j$.*

proof of claim 10: To any random oracle query X, the relay algorithm R_0 always responds with the random oracle output $F(X)$. Thus the situation described in the statement of the claim occurs if and only if the simulator responds to its queries (corresponding to the κ-bit blocks in $g(X) = x_1 \parallel \ldots \parallel x_j$) in such a way that $y_j \oplus z_j \neq F(X)$.

From claim 9, we can deduce that if the distinguisher is to compute the Davies-Meyer Merkle Damgård output on $g(X) = x_1 \parallel \ldots \parallel x_j$, then the only way to do this is to make the ordered sequence of queries $(+, x_1, y_1), \ldots, (+, x_j, y_j)$ unless the simulator S_0 fails. Here $y_1 = IV$ and for each $i = 2 \ldots j$ we have $y_i = y_{i-1} \oplus z_{i-1}$. Hence the simulator S_0 already has the triples $(x_1, y_1, z_1) \ldots (x_{j-1}, y_{j-1}, z_{j-1})$ in its table $\mathcal{T}$ when the query $(+, x_j, y_j)$ is made.

If the response of the simulator S_0 to the query $(+, x_j, y_j)$ is different from $F(X) \oplus y_j$, then it must be the case that the simulator is unable to give this response because of some other constraint. But from claim 8, we can deduce that the block x_j cannot be part of any other valid Davies-Meyer Merkle-Damgård computation sequence unless the simulator S_0 fails. Thus there can be no other constraint of the response of S_0 if it has not explicitly failed.

Thus the responses of S_0 are always consistent with the relay algorithm R_0 in game 3, if it does not fail.

□

In fact, we can use the same argument as in proof of claim 10 to show that the responses of S_0 are consistent with the random oracle F in game 4 as well (that is, the result of applying Davies-Meyer Merkle-Damgård construction using S_0 to $g(X)$ is the same as $F(X)$).

From the above, we can deduce that if the simulator S_0 does not fail in game 4, then the responses of the relay algorithm R_1 are identical to the responses of the relay algorithm R_0. And since we are using the same simulator S_0 in both games, and have shown that the responses of the simulator and the two relay algorithms are consistent in the two games, we can also deduce that the view of the distinguisher D remains unchanged from game 3 to game 4 if the simulator S_0 does not fail in either of the two games.

Hence, we can finally complete the proof of claim 7 by observing that if the maximum length of the prefix-free encoding of a random oracle query made by D is $\ell \cdot \kappa$ then,

$$\begin{aligned} |\Pr[G_4] - \Pr[G_3]| &\leq \Pr\left[S_0 \text{ fails in game 3}\right] + \Pr\left[S_0 \text{ fails in game 4}\right] \\ &= \mathcal{O}\left(\frac{(q_E + q_F\ell)^2}{2^n}\right) \\ &= \mathcal{O}\left(\frac{(q\ell)^2}{2^n}\right) \end{aligned}$$

□

Game 5. In this game, we modify the simulator S_0 so as to make its responses independent of the random oracle F. For this purpose, we remove the random oracle F from this game entirely and the new simulator S_1 always chooses a random n-bit response, even in situations where S_0 would have consulted the RO F. We also remove all the failure conditions from the new simulator S_1.

Thus on a forward query $(+, x, y)$, the new simulator S_1 checks if there is a triple (x, y, z) in its table $\mathcal{T}$. If it finds such a triple then it responds with the n-bit string z. Otherwise it chooses a uniformly random n-bit string z and sends this as its response, while storing the triple (x, y, z) in $\mathcal{T}$. On an inverse query $(-, x, z)$, it similarly checks to see if there is a triple (x, y, z) in its table $\mathcal{T}$. If it finds such a triple, it responds with y, else it chooses a uniformly random n-bit response y.

Now we will show that the view of the distinguisher D does not change by a non-negligible amount from game 4 to game 5. In fact, if we can show that the responses of the simulators S_0 and S_1 seem almost identical to the distinguisher D, then we will be done. But the responses of these two simulators are identical apart from the failure conditions which are used by S_0 and not by S_1 (even when S_0 consults the random oracle, its response is still uniformly distributed). Thus, the distinguisher does not notice a difference between these games if:

- In game 4, the simulator S_0 does not fail.

- In game 5, the simulator S_1 does not respond to its queries in such a manner that its satisfy one of the failure conditions specified in the definition of S_0.

In fact, these two events are identical in terms of their probability of occurrence since the distribution of the responses of the two simulators is identical. Let G_5 denote the event that the distinguisher D outputs 1 in game 5, so that $\Pr[G_5] = \Pr[D^{R_1^{S_1}, S_1}(1^\lambda) = 1]$. Then we can deduce that,

$$\begin{aligned} |\Pr[G_5] - Pr[G_4]| &\leq \Pr\left[S_0 \text{ fails in game 4}\right] \\ &\quad + \Pr\left[S_1 \text{ should have failed in game 5}\right] \\ &= \mathcal{O}\left(\frac{q^2\ell^2}{2^n}\right) \end{aligned}$$

Game 6. This is the final game of our argument. Here we finally replace the simulator S_1 with the ideal cipher E. Since the relay algorithm R_1 simply implemented the construction pf-MD$_g^{S_1}$, it will be the same as the RO construction pf-MD$_g^E$ in this game. Hence this game is same as the view of the distinguisher in the *ideal cipher model*.

The outputs of the ideal cipher E are not distributed uniformly like the responses of S_1. Hence the distinguisher may be able to differentiate between games 5 and 6 if it can detect this. However, this happens only if S_1 outputs an input/output collision for the same ideal cipher key. The probability of this event is easily seen to be at most the birthday bound. Let G_6 denote the probability that the distinguisher outputs 1 in game 6, so that $\Pr[G_6] =$

$\Pr[D^{\text{pf-MD}_g^E,E}(1^\lambda) = 1]$. Then we can deduce that

$$|\Pr[G_6] - \Pr[G_5]| = \mathcal{O}\left(\frac{q^2\ell^2}{2^n}\right)$$

Now we can complete the proof of lemma 1 by combining games 1 to 6, and observing that game 1 is same as the random oracle model while game 6 is same as the ideal cipher model. Hence we can deduce that

$$\left|\Pr\left[D^{F,S^F}(1^\lambda) = 1\right] - \Pr\left[D^{\text{pf-MD}_g^E,E}(1^\lambda) = 1\right]\right| = \mathcal{O}\left(\frac{q^2\ell^2}{2^n}\right)$$

□

3.2.2 MD-then-Chop Construction

Now we will prove the indifferentiability of the second random oracle construction chop-MD_s^E. Recall that this construction essentially applies the plain Davies-Meyer Merkle-Damgård construction (using the ideal cipher E) to the input and then removes a non-trivial number s of the output bits.

Lemma 2. *The Merkle-Damgård construction with truncated output chop-MD_s^E based on the Davies-Meyer construction applied to an ideal cipher $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ is (t_D, t_S, q, ϵ)-indifferentiable from a random oracle $F : \{0,1\}^* \to \{0,1\}^{n-s}$ in the ideal cipher model for E, for any t_D and $t_S = \mathcal{O}(q^2 \cdot \kappa))$, with $\epsilon = 2^{-n} \cdot \ell^2 \cdot \mathcal{O}(q^2)$.*

Proof:

We will assume that the random oracle inputs provided to the construction chop-MD_s^E are all of length, that is a multiple of the block length κ. In actual implementation, this can be achieved by applying an appropriate encoding scheme to the input, such as appending a 1 followed by a sufficient number of 0s to the input.

The Simulator. The simulator S accepts either forward ideal cipher queries, $(+, x, y)$, or inverse ideal cipher queries, $(-, x, z)$, such that $x \in \{0,1\}^\kappa$ and $y, z \in \{0,1\}^n$. In either case, the simulator responds with a n-bit string that is interpreted as $E_x(y)$ in case of a forward query $(+, x, y)$, and as $E_x^{-1}(z)$ in case of an inverse query $(-, x, z)$. The simulator maintains a table $\mathcal{T}$ consisting of triples $(x, y, z) \in \{0,1\}^\kappa \times \{0,1\}^n \times \{0,1\}^n$, such that it either responded with z to a forward query $(+, x, y)$ or with y to an inverse query $(-, x, z)$.

On getting a forward query $(+, x, y)$, the simulator searches its table $\mathcal{T}$ for a triple of the form (x, y, z). If it finds such a triple then it responds with the n-bit string z otherwise it needs to choose a fresh response to this query. It proceeds by searching its table $\mathcal{T}$ for a sequence of triples $(x_1, y_1, z_1) \ldots (x_i, y_i, z_i)$ such that:

- It is the case that $y_1 = IV$, where IV denotes the initialization vector used in the construction chop-MD_s^E.

- For each $j = 2 \ldots i$, it holds that $y_j = y_{j-1} \oplus z_{j-1}$.
- It is the case that $y = y_i \oplus z_i$, where y is the ideal cipher input from the current forward query.

Note that for an empty sequence of triples, i.e. when just considering the κ-bit block x from the current query, we only need to check if $y = IV$ and none of the above conditions make sense.

If the simulator finds such a sequence of triples, then it needs to give a response that is consistent with the random oracle output on $x_1 \parallel \ldots \parallel x_i \parallel x$. Thus, the simulator makes this RO query to get the output $Y = F(x_1 \parallel \ldots \parallel x_i \parallel x)$. It then samples a uniformly random s-bit string Y' and outputs the n-bit string $z = (Y \parallel Y') \oplus y$. If the simulator does not find any such sequence of triples in its table $\mathcal{T}$, then it samples a uniformly distributed random n-bit string z and sends z as its response. In either case, it inserts the triple (x, y, z) in its table $\mathcal{T}$.

On an inverse query $(-, x, z)$, the simulator S searches its table $\mathcal{T}$ for a triple (x, y, z) with arbitrary y. If it finds such a triple, then it responds with y. Otherwise, the simulator S chooses a uniformly distributed random n-bit string y and responds with y. It then inserts the triple (x, y, z) in its table $\mathcal{T}$.

Proof of Indifferentiability. We need to prove that the distinguisher cannot tell apart the two scenarios, one where it has oracle access to the random oracle F and the simulator S, and the other where it has oracle access to the RO construction chop-MD_s^E and the ideal cipher E. As in the case of the prefix-free Merkle-Damgård construction, the proof involves a hybrid argument.

Game 1. This is the *random oracle model*, and the distinguisher D is given oracle access to the random oracle F and the ideal cipher simulator S described above. Let G_1 denote the event that the distinguisher D outputs 1 in this game.

$$\Pr[G_1] = \Pr[D^{F,S^F}(1^\lambda) = 1]$$

Game 2. In this game, the distinguisher is given oracle access to a *relay algorithm* R_0 instead of direct oracle access to F. The *relay algorithm*, in turn, has oracle access to the random oracle F. On a random oracle query X, the relay algorithm simply makes the same query to F and responds with the RO output $F(X)$. Let G_2 denote the event that D outputs 1 in game 2. Since the view of the distinguisher remains unchanged in this game, we can deduce that

$$\Pr[G_2] = \Pr[D^{R_0^F,S^F}(1^\lambda) = 1] = \Pr[G_1]$$

Game 3. In this game, we modify the simulator S. In particular, we restrict the responses of the simulator such that they never satisfy certain specific failure conditions. If the simulator comes up with a response that results in its responses satisfying one of these conditions, then it explicitly fails instead of sending this response.

These failure conditions, that the new simulator S_0 checks for, describe certain dependencies among its responses that could be exploited by a distinguisher. In response to a forward query $(+, x, y)$, the new simulator S_0 starts by choosing a n-bit response $z \in \{0,1\}^n$ in the same way as the original simulator S. It then checks if one of the following conditions is satisfied:

1. *Condition B_1:* It is the case that $z \oplus y = IV$, where IV is the initialization vector used in the RO construction chop-MD_s^E.

2. *Condition B_2:* There is a triple $(x', y', z') \in \mathcal{T}$, with $(x', y') \neq (x, y)$, such that $y' \oplus z' = y \oplus z$.

3. *Condition B_3:* There is a triple $(x', y', z') \in \mathcal{T}$, with $(x', y') \neq (x, y)$, such that $y \oplus z = y'$.

If the response z, whether S_0 chooses a uniformly random z or z is chosen to be consistent with the RO F on some query, is such that one of these conditions is satisfied, then the simulator S_0 explicitly fails.

On a new inverse query $(-, x, z)$, the simulator S_0 again chooses its response $y \in \{0,1\}^n$ in the same way as S. It then checks if the following conditions, and fails if any one of them is satisfied:

1. *Condition C_1:* It is the case that $y = IV$ or $y \oplus z = IV$, where IV is the initialization vector used in the RO construction chop-MD_s^E.

2. *Condition C_2:* There is a triple $(x', y', z') \in \mathcal{T}$, with $(x', z') \neq (x, z)$, such that $y' \oplus z' = y \oplus z$.

3. *Condition C_3:* There is a triple $(x', y', z') \in \mathcal{T}$, with $(x', z') \neq (x, z)$, such that either $y \oplus z = y'$ or $y' \oplus z' = y$.

Next we will estimate the occurrence probability for each of the above failure conditions. We start by noting that the probability that one of the conditions C_1, C_2 and C_3 holds can be readily estimated, since the simulator always chooses uniformly random responses to inverse queries.

In the case of a forward query, the simulator might be forced to choose its response so as to maintain consistency with the random oracle F. Hence the distinguisher could find out $(n - s)$ bits of the response of the simulator by making a random oracle query. Thus, it is not as straightforward to estimate the occurrence probabilities for the failure conditions B_1, B_2 and B_3. Let the number of random oracle queries made by D be q_F, and let the number of ideal cipher queries be q_E (hence the total number of queries $q = q_E + q_F$)

We can bound the occurrence probability of event B_1 easily, since it is the probability that at least one of $(q_E + q_F)$ uniformly random n-bit strings is IV. In order to estimate the occurrence probability of failure condition B_2, we will analyze three situations separately.

- Query $(+, x, y)$ does not correspond to the last block of a random oracle input. In this case, condition B_2 holds only if the uniformly random n-bit string $y \oplus z$ is equal to one of upto q_E n-bit strings corresponding to previous queries.

- Both (x, y, z) and (x', y', z') correspond to last blocks of random oracle inputs, and the simulator adjusted its response according to the RO output in each case. In this case, condition B_2 implies a collision among the two random oracle outputs as well as a collision among the remaining s uniformly random bits chosen by the simulator in each case.

- The triple (x, y, z) forms the last block of a random oracle input and the simulator adjusts its response z accordingly, but (x', y', z') does not. In this case, $y' \oplus z'$ is a random n-bit string chosen by the simulator. Here, the condition B_2 corresponds to a random oracle output along with the extra s random bits chosen by the simulator colliding with another randomly and independently chosen n-bit string chosen by the simulator.

From the above, we can deduce that the occurrence probability of failure condition B_2 can be bounded by the birthday bound over $(q_E + q_F)$ random n-bit strings.

In order to bound the occurrence probability of failure condition B_3, we note that the simulator S_0 chooses at least s random and independent bits in its response (even if it is forced to make the remaining $(n - s)$ bits consistent with the random oracle). Thus the occurrence probability of condition B_3 can be bounded by the birthday bound over $(q_E + q_F)$ independent and random s-bit random strings.

Let G_3 denote the event that the distinguisher D outputs 1 in this game, i.e. $\Pr[G_3] = \Pr\left[D^{R_0^F, S_0^F}(1^\lambda) = 1\right]$. The responses of the distinguisher in games 2 and 3 differ only if the simulator S_0 exits because of one of the failure conditions in game 3. This event is identical with the event that at least one of the failure conditions hold for the responses of either simulators (in which case S_0 exits while S does not).

$$\begin{aligned} |\Pr[G_3] - \Pr[G_2]| &\leq \Pr[B_1 \cup B_2 \cup B_3 \cup C_1 \cup C_2 \cup C_3 \text{ hold for a query.}] \\ &= \mathcal{O}\left(\frac{q^2}{2^s}\right) \end{aligned}$$

Game 4. In this game, we modify the relay algorithm but leave the ideal cipher simulator S_0 unchanged. The underlying idea is to make the responses of the relay algorithm directly dependent on the simulator. Thus, instead of giving the new relay algorithm R_1 oracle access to the random oracle F, here it is given oracle access to the simulator S_0. It responds to a random oracle query X by computing the Davies-Meyer Merkle-Damgård construction using input X and then chops the same s bits from the output as in the case of the RO construction chop-MD_s^E.

Let G_4 denote the event that the distinguisher D outputs 1 in game 4. Thus we know that

$$\Pr[G_4] = \Pr\left[D^{R_1^{S_0}, S_0^F}(1^\lambda) = 1\right]$$

We will assume that the maximum length of a random oracle query made by the adversary is $\ell \cdot \kappa$. Now we will show that the view of the distinguisher changes by at most a negligible amount in the transition from game 3 to game 4. This claim is formally stated below.

Claim 11. *Let G_3 and G_4 denote the events that the distinguisher outputs 1 in game 3 and game 4, respectively. Let q_E and q_F denote the number of ideal cipher and random oracle queries made by the distinguisher, then it is the case that*

$$|\Pr[G_4] - \Pr[G_3]| = \mathcal{O}\left(\frac{(q_E + q_F \cdot \ell)^2}{2^s}\right)$$

proof of claim 11: The view of the distinguisher differs in games 3 and 4 only if it finds a difference in responses of either the relay algorithm or the simulator among the two games. We will show that such a difference is impossible, unless the simulator S_0 fails in at least one of the two games. Let us start by proving a few important properties of the simulator S_0 that are valid in both games 3 and 4.

Claim 12. *If the simulator S_0 does not explicitly fail, then there are no two different sequences of κ-bit blocks $x_1 \dots x_m$ and $x'_1 \dots x'_p$ with corresponding triples $(x_1, y_1, z_1) \dots (x_m, y_m, z_m)$ and $(x'_1, y'_1, z'_1) \dots (x'_p, y'_p, z'_p)$ in the table $\mathcal{T}$ such that:*

- *It is the case that $y_1 = y'_1 = IV$, and for each $b = 2 \dots m$ and $b' = 2 \dots p$, it holds that $y_b = y_{b-1} \oplus z_{b-1}$ and $y'_{b'} = y'_{b'-1} \oplus z'_{b'-1}$.*
- *It is the case that $(x_m, y_m, z_m) = (x'_p, y'_p, z'_p)$.*

proof of claim 12: We will prove this claim by performing an induction on the number of queries made to the simulator and show that unless the simulator S_0 fails, such sequences of triples cannot exist. When no queries have been made as yet, this claim is vacuously true. Let us assume that the claim is also true when q queries have been made to the simulator S_0.

Now say there exist two sequences of triples be $(x_1, y_1, z_1) \ldots (x_m, y_m, z_m)$ and $(x'_1, y'_1, z'_1) \ldots (x'_p, y'_p, z'_p)$, that satisfy the properties stated in the claim, after the $(q+1)^{th}$ query. Since we know that $(x_m, y_m, z_m) = (x'_p, y'_p, z'_p)$, we can deduce that there are two subsequences of κ-bit blocks $x_{m-r} \ldots x_m$ and $x_{p-r} \ldots x_p$ such that

$$\forall b \in \{0, r\} \; : \; (x_{m-b}, y_{m-b}, z_{m-b}) = (x'_{p-b}, y'_{p-b}, z'_{p-b})$$

If $r < m-1$ and $r < p-1$, then consider the triples $(x_{m-r-1}, y_{m-r-1}, z_{m-r-1})$ and $(x'_{p-r-1}, y'_{p-r-1}, z'_{p-r-1})$. Since $y_{m-r} = y'_{p-r}$, we can deduce that $y_{m-r-1} \oplus z_{m-r-1} = y'_{p-r-1} \oplus z'_{p-r-1}$. Without loss of generality, assume that the query corresponding to the triple $(x_{m-r-1}, y_{m-r-1}, z_{m-r-1})$ was made earlier than the one corresponding to $(x'_{p-r-1}, y'_{p-r-1}, z'_{p-r-1})$. If this query is a forward query, then the simulator S_0 would fail because of failure condition B_2. On the other hand, if this were an inverse query, then the simulator would have failed due to failure condition C_2.

Now consider the case that $r = p-1$ but $r < m-1$. In this case, if the triple $(x_{m-r-1}, y_{m-r-1}, z_{m-r-1})$ was generated as a result of a forward query then the simulator S_0 would have failed due to failure condition B_1 because $y_{m-r-1} \oplus z_{m-r-1} = y_{m-r} = y'_1 = IV$. If this triple were generated as a result of an inverse query then the simulator would have failed as a result of failure condition C_1 being true. The case when $r = m - 1$ but $r < p - 1$ is symmetrical

Lastly, it cannot be the case that $r = p - 1$ as well as $r = m - 1$, since the two bit strings $x'_1 \parallel \ldots \parallel x'_p$ and $x_1 \parallel \ldots \parallel x_m$ are different.

Hence we can conclude that there can be no such sequences of κ-bit blocks $x_1, \ldots, x_m$ and $x_1, \ldots, x'_p$ if the simulator does not explicitly fail. $\square$

Next we show that if the distinguisher wishes to find the random oracle output for an input $X = x_1 \parallel \ldots \parallel x_s$ by making queries to the simulator S_0 and computing the Davies-Meyer Merkle-Damgård construction, then the only way it can do so is by making the ordered sequence of forward queries $(+, x_1, y_1) \ldots (+, x_s, y_s)$.

Claim 13. *Consider any sequence of κ-bit blocks $x_1 \ldots x_s$, with corresponding triples $(x_1, y_1, z_1) \ldots (x_s, y_s, z_s)$ in the table $\mathcal{T}$ maintained by the simulator S_0, such that $y_1 = IV$ and for each $j = 2 \ldots s$ it holds that $y_j = y_{j-1} \oplus z_{j-1}$. If the simulator S_0 does not explicitly fail then it must be the case that the triples $(x_1, y_1, z_1) \ldots (x_s, y_s, z_s)$ are generated as a result of the ordered sequence of forward queries $(+, x_1, y_1) \ldots (+, x_s, y_s)$.*

proof of claim 13: To the contrary, assume that the triples $(x_1, y_1, z_1) \ldots$

(x_s, y_s, z_s) were not generated as a result of the sequence of forward queries mentioned in the claim. We can then deduce that one of the following must be true regarding the actual sequence of queries that resulted in these triples:

1. For $j = 1 \ldots (s-1)$, a forward query $(+, x_j, y_j)$ was made when the triple $(x_{j+1}, y_{j+1}, z_{j+1})$ already existed in the table $\mathcal{T}$.

2. For $j = 2 \ldots s$, an inverse query $(-, x_j, z_j)$ was made when the triple $(x_{j-1}, y_{j-1}, z_{j-1})$ already existed in the table $\mathcal{T}$.

3. The triple (x_1, y_1, z_1) was generated as a result of an inverse query $(-, x_1, y_1)$.

In the first case, the simulator S_0 would fail since the failure condition B_3 holds. Indeed, we can deduce that $y_j \oplus z_j = y_{j+1}$. In the second case, the simulator explicitly fails because of failure condition C_3 since we know that $y_j = y_{j-1} \oplus z_{j-1}$. In the third and final case, the simulator would explicitly fail since the failure condition C_1 holds. Thus the only possible sequence of queries that could result in these triples is the one mentioned in the claim. $\square$

Now we will show that the responses of the relay algorithm R_0 in game 3 are consistent with those of the simulator S_0. Note that in game 4, the relay algorithm R_1 is designed in such a way that its responses are always consistent with S_0 while the relay algorithm R_0 is given oracle access to the random oracle F and may not be consistent with S_0. We show that such inconsistency is impossible unless the simulator S_0 explicitly fails.

Claim 14. *In game 3, if the simulator S_0 never fails then there is no sequence of κ-bit blocks $x_1 \ldots x_j$, with corresponding triples $(x_1, y_1, z_1) \ldots (x_j, y_j, z_j)$ such that:*

- *$y_1 = IV$ and for $l = 2 \ldots j$ it is the case that $y_l = y_{l-1} \oplus z_{l-1}$.*
- *To the random oracle query $X = x_1 \parallel \ldots \parallel x_j$, the response of the relay algorithm R_0 is different from the $(n - s)$ bits of $y_j \oplus z_j$ that are not chopped in the construction chop-MD_s^E.*

proof of claim 14: To any random oracle query X, the relay algorithm R_0 always responds with the random oracle output $F(X)$. Thus the situation described in the statement of the claim occurs if and only if the simulator responds to its queries (corresponding to the κ-bit blocks in $X = x_1 \parallel \ldots \parallel x_j$) in such a way that $y_j \oplus z_j \neq F(X)$.

From claim 13, we can deduce that if the distinguisher is to compute the RO output on $X = x_1 \parallel \ldots x_j$ by querying the simulator, then the only way to do this is to make the ordered sequence of queries $(+, x_1, y_1), \ldots, (+, x_j, y_j)$ unless the simulator S_0 fails. Here $y_1 = IV$ and for each $i = 2 \ldots j$ we have $y_i = y_{i-1} \oplus z_{i-1}$. Hence the simulator S_0 already has the triples $(x_1, y_1, z_1) \ldots (x_{j-1}, y_{j-1}, z_{j-1})$ in its table $\mathcal{T}$ when the query $(+, x_j, y_j)$ is made.

If the response of the simulator S_0 to the query $(+, x_j, y_j)$ is different from $F(X) \oplus y_j$, then it must be the case that the simulator is unable to give this response because of some other constraint. But from claim 12, we can deduce that the block x_j cannot be the last block of any other valid Davies-Meyer Merkle-Damgård computation sequence unless the simulator S_0 fails. Thus there can be no other constraint of the response of S_0 if it has not explicitly failed. □

Thus we have shown that, even though the relay algorithm R_0 simply forwards the random oracle outputs in game 3, its responses are still consistent with the responses of simulator S_0 in that game. Another way to look at this claim would be to note that the responses of the simulator S_0 are always consistent with the random oracle outputs, unless it explicitly fails.

Hence, it is easy to see that if the simulator S_0 does not fail in either of the games 3 or 4, the view of the distinguisher does not change in going from one game to the other. Now we can complete the proof of claim 11 by observing that if the longest RO query made by the distinguisher D consists consists of at most ℓ κ-bit blocks then

$$\begin{aligned} |\Pr[G_4] - \Pr[G_3]| &\leq \Pr[S_0 \text{ fails in game 3}] + \Pr[S_0 \text{ fails in game 4}] \\ &= \mathcal{O}\left(\frac{(q_E + q_F \cdot \ell)^2}{2^s}\right) \\ &= \mathcal{O}\left(\frac{(q \cdot \ell)^2}{2^s}\right) \end{aligned}$$

□

Game 5. In this game, we modify the simulator S_0 so as to make the view of the distinguisher independent of the random oracle F. For this purpose, we introduce a new simulator S_1 that does not have oracle access to the random oracle F, and always outputs a n-bit random response to all new forward as well as inverse queries even in cases where S_0 would have maintained consistency with F. We also remove all failure conditions from the simulator S_1.

On a forward query $(+, x, y)$, the new simulator S_1 checks if there already exists a triple (x, y, z) in its table $\mathcal{T}$. If it finds such a triple, then it responds with the n-bit string z. If not, then it chooses a uniformly random n-bit string z and sends this as its response, while storing the triple (x, y, z) in $\mathcal{T}$. On an inverse query $(-, x, z)$, it similarly checks to see if there is a triple (x, y, z) in its table $\mathcal{T}$. If it finds such a triple, it responds with y otherwise it chooses a uniformly random n-bit response y.

Now we will show that the view of the distinguisher does not change by a non-negligible amount in going from game 4 to game 5. Note that if we can show that the responses of the simulators S_0 and S_1 are indistinguishable, then we will be done. But in the view of the distinguisher, these two simulators are identical apart from the failure conditions used by S_0 but not by S_1. Thus, we can deduce that the distinguisher does not notice a difference between games 4 and 5 unless:

- In game 4, simulator S_0 explicitly fails.
- In game 5, simulator S_1 responds with an output such that it satisfies one of the failure conditions (for which S_0 would have failed).

Since the simulator S_1 always chooses a uniformly random n-bit response to every query, we can easily bound the occurrence probability of any of the failure conditions using the birthday bound. Let G_5 denote the event that the distinguisher D outputs 1 in game 5, so that $\Pr[G_5] = \Pr[D^{R_1^{S_1},S_1}(1^\lambda) = 1]$. Thus we can deduce that

$$\begin{aligned}
|\Pr[G_5] - \Pr[G_4]| &\leq \Pr[S_0 \text{ fails in game 4}] \\
&\quad + \Pr[S_1 \text{ satisfies a failure condition in game 5}] \\
&= \mathcal{O}\left(\frac{(q\cdot\ell)^2}{2^s} + \frac{(q\cdot\ell)^2}{2^n}\right) \\
&= \mathcal{O}\left(\frac{q^2\ell^2}{2^s}\right)
\end{aligned}$$

Game 6. This is the final game of our proof. In this game, we replace the simulator S_1 with the ideal cipher E. Since the relay algorithm R_1 essentially implements the RO construction chop-MD_s^E, the view of the distinguisher in this game is essentially its view in the *ideal cipher model*.

The outputs of the ideal cipher E are not uniformly distributed as are the responses of S_1. However, the distinguisher can differentiate between the two only if the simulator S_1 outputs a collision for the same ideal cipher key. The occurrence probability of this event can be easily bounded using the birthday bound. Thus let G_6 be the event that the distinguisher D outputs 1 in this game, so that $\Pr[G_6] = \Pr[D^{\text{chop-MD}_s^E, E}(1^\lambda) = 1]$ and we can deduce that

$$|\Pr[G_5] - \Pr[G_4]| \leq \mathcal{O}\left(\frac{q^2\ell^2}{2^n}\right)$$

Now we can complete the proof of lemma 2 by combining games 1 to 6, and observing that game 1 is same as the random oracle model while game 6 is the same as the ideal cipher model. Hence we can deduce that

$$\left|\Pr\left[D^{F,S^F}(1^\lambda) = 1\right] - \Pr\left[D^{\text{chop-MD}_s^E, E}(1^\lambda) = 1\right]\right| = \mathcal{O}\left(\frac{q^2\ell^2}{2^n}\right)$$

$\square$

3.2.3 NMAC and HMAC Constructions

Here we will prove the indifferentiability of the NMAC and HMAC construction with the Davies-Meyer compression function.

Lemma 3. *The NMAC construction $NMAC^{E1,E2}$ that uses two independent ideal block ciphers $E1 : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ and $E2 : \{0,1\}^{\kappa'} \times \{0,1\}^{n'} \to \{0,1\}^{n'}$ is (t_D, t_S, q, ϵ)-indifferentiable from a random oracle $F : \{0,1\}^* \to \{0,1\}^{n'}$ in the ideal block cipher model for $E1$ and $E2$, for any t_D and $t_S = \mathcal{O}(q^2)$, with $\epsilon = 2^{-\min(n,n')} \cdot \ell^2 \cdot \mathcal{O}(q^2)$ ($\ell\kappa$ is the maximum length of an RO query made by the distinguisher).*

Proof: Recall that the construction $\mathrm{NMAC}^{E1,E2}$ essentially applies the Davies-Meyer Merkle-Damgård construction using the block cipher $E1$ to the input $x_1 \parallel \ldots \parallel x_\ell$ to get the final output Y. It then applies the Davies-Meyer compression function using $E2$ to this output Y. We will assume for simplicity that the output length n of $E1$ is the same as the key length κ' of $E2$[1]. We will use the initialization vector IV for the Davies-Meyer Merkle-Damgård construction applied to $E1$, and use initialization vector IV' for the Davies-Meyer construction with $E2$.

The Simulator. Let us start by describing the simulator for the ideal block ciphers $E1$ and $E2$ in the random oracle model with an actual random oracle F. The simulator gets forward/inverse queries for either of the block ciphers $E1$ and $E2$. Thus the queries that simulator S responds to are as follows:

1. $(1, +, x, y)$: A forward $E1$ query, where $(x, y) \in \{0,1\}^\kappa \times \{0,1\}^n$. The expected response is $E1_x(y)$.

2. $(1, -, x, z)$: An inverse $E1$ query, where $(x, z) \in \{0,1\}^\kappa \times \{0,1\}^n$. The expected response is $E1_x^{-1}(z)$.

3. $(2, +, x, y)$: A forward $E2$ query, where $(x, y) \in \{0,1\}^{\kappa'} \times \{0,1\}^{n'}$. The expected response is $E2_x(y)$.

[1]one can use suitable padding techniques to expand Y from n bits to κ' bits

4. $(2, -, x, z)$: An inverse $E2$ query, where $(x, z) \in \{0,1\}^{\kappa'} \times \{0,1\}^{n'}$. The expected response is $E2_x^{-1}(z)$.

The simulator S also maintains a table $\mathcal{T}$ in which it records all previous queries that were made to it, along with the responses it gave to each. Thus, it records an entry $(1, x, y, z)$ in $\mathcal{T}$ for every forward (resp. inverse) query of the form $(1, +, x, y)$ (resp. $(1, -, x, z)$) to which it responded with z (resp. y). On the other hand, it records an entry $(2, x, y, z)$ in $\mathcal{T}$ for every forward (resp. inverse) query of the form $(2, +, x, y)$ (resp. $(2, -, x, z)$) to which it responded with z (resp. y).

On getting a forward query $(1, +, x, y)$, the simulator first checks if there is a tuple $(1, x, y, z)$ in its table $\mathcal{T}$. If this is the case, then the simulator S responds with z, otherwise it chooses a uniformly random n-bit string z and sends this as its response. It then records $(1, x, y, z)$ in its table $\mathcal{T}$.

Similarly, on getting an inverse query $(1, -, x, z)$, it first searches its table $\mathcal{T}$ for a tuple $(1, x, y, z)$. If it finds such a tuple, then it responds with z, otherwise it sends a uniformly random n-bit string y as its response and stores $(1, x, y, z)$ in its table $\mathcal{T}$.

On a query $(2, +, x, y)$, the simulator S again checks if there is a tuple $(2, x, y, z) \in \mathcal{T}$. If this is the case then it responds with z. If it cannot find such a tuple, then the simulator checks if $y = IV'$, where IV' is the initialization vector used in the second part of the construction $\text{NMAC}^{E1,E2}$. If $y \neq IV'$, then the simulator simply sends back a random response $z \in \{0,1\}^{n'}$ and stores $(2, x, y, z)$ in $\mathcal{T}$. On the other hand, if $y = IV'$, then the simulator

S searches its table $\mathcal{T}$ for a sequence of tuples $(1, x_1, y_1, z_1), \ldots, (1, x_i, y_i, z_i)$ such that the following conditions hold:

- It is the case that $y_1 = IV$, where IV denotes the initialization vector used in $\text{NMAC}^{E1,E2}$.
- For each $j = 2 \ldots i$, it holds that $y_j = y_{j-1} \oplus z_{j-1}$.
- It is the case that $y_i \oplus z_i = x$, where x is the key provided in the current query $(2, +, x, y)$ (here we assume that $\kappa' = n$).

If the simulator S finds such a sequence of tuples, then it needs to send a response that is consistent with the random oracle F. Thus, it queries the random oracle F on the input $x_1 \parallel \ldots \parallel x_\ell$ to get the output $Y = F(x_1$ $parallel \ldots \parallel x_\ell)$. It then chooses its response as $z = Y \oplus y = Y \oplus IV'$ (since we know that $y = IV'$). It then sends this n'-bit string z as its response and store $(2, x, y, z)$ in its table $\mathcal{T}$. If S does not find such a tuple, then it sends a random response $z \in \{0, 1\}^{n'}$ and stores $(2, x, y, z)$ in $\mathcal{T}$.

On getting an inverse query $(2, -, x, z)$, the simulator searches its table $\mathcal{T}$ for a tuple $(2, x, y, z)$ and responds with y if it finds such a tuple. If it does not find such a tuple, then it sends a uniformly random n'-bit response y and stores $(2, x, y, z)$ in its table $\mathcal{T}$.

Proof of Indifferentiability. We need to show that the distinguisher cannot tell apart the two scenarios, one where it has oracle access to the actual random oracle F and the simulator S described above, and the other where it has oracle access to RO construction $\mathrm{NMAC}^{E1,E2}$ and the ideal block ciphers $E1$ and $E2$. We will use a hybrid argument to prove this result starting in the random oracle scenario, and ending in the ideal cipher scenario through a sequence of indistinguishable games.

Game 1. This is the random oracle model, where the distinguisher D has oracle access to the random oracle F and the simulator S. Let G_1 denote the event that D outputs 1 after interacting with F and S. Thus,

$$\Pr[G_1] = \Pr\left[D^{F,S^F}(1^\lambda) = 1\right]$$

Game 2. In this game, we give the distinguisher oracle access to a dummy *relay algorithm* R_0 instead of direct oracle access to the RO F. This relay algorithm, in turn, has oracle access to the RO F, and on getting a random oracle query from the distinguisher, it simply makes the same query to F and forwards the RO output to the distinguisher D as its response. The simulator S still has direct oracle access to F. Let G_2 denote the event that the distinguisher D outputs 1 in this game. Since the view of the distinguisher remains unchanged in this game, we can deduce that

$$\Pr[G_2] = \Pr\left[D^{R_0^F,S^F}(1^\lambda) = 1\right] = \Pr[G_1]$$

Game 3. In this game, we will modify the simulator S by restricting its responses. In particular, the new simulator S_0 chooses its responses in the same fashion as the original simulator S, but after making its choice the simulator S_0 checks if it responses so far satisfy one of a few conditions that could aid the distinguisher in getting to know that it is in the random oracle scenario.

On a forward query $(1, +, x, y)$, the new simulator S_0 checks if there is a tuple $(1, x, y, z)$ in its table $\mathcal{T}$, and chooses its response z in the same way as the original simulator S. However, if the response chosen is a new one then it checks if the tuple (x, y, z) satisfies one of the following conditions before sending z.

1. *Condition B_1:* It is the case that $z \oplus y = IV$, where IV is the n-bit initialization vector used in the first Merkle-Damgård construction using $E1$.

2. *Condition B_2:* There is a tuple $(1, x', y', z') \in \mathcal{T}$, with $(x', y') \neq (x, y)$, such that $y' \oplus z' = y \oplus z$.

3. *Condition B_3:* There is a tuple $(1, x', y', z') \in \mathcal{T}$ such that $z \oplus y = y'$.

4. *Condition B_4:* There is a tuple $(2, x', y', z') \in \mathcal{T}$ such that $y \oplus z = x'$.

If the response z chosen by the simulator S_0 is such that at least one of these conditions is satisfied, then the simulator explicitly fails. Essentially, the idea is that conditions B_1 and B_2 could be used by the distinguisher to make two random oracle inputs collide after the Merkle-Damgård part using $E1$. On the other hand, conditions B_3 and B_4 could be used by the distinguisher to generate a random oracle input such that the simulator cannot adjust its output to match that of the random oracle. Since the simulator S_0 always chooses the response to any $E1$ query at random, we can bound the occurrence probabilities of each of these events using simple probability calculations.

On an inverse query $(1, -, x, z)$, the new simulator S_0 chooses its response y in the same fashion as the original simulator S. However, if the response is not chosen from the table $\mathcal{T}$, then S_0 checks if the tuple (x, y, z) satisfies any of the following conditions.

1. *Condition C_1:* It is the case that $y = IV$ or $y \oplus z = IV$, where IV is the initialization vector used in the Merkle-Damgård construction using $E1$.

2. *Condition C_2:* There is a tuple $(1, x', y', z') \in \mathcal{T}$, with $(x', z') \neq (x, z)$, such that $y' \oplus z' = y \oplus z$.

3. *Condition C_3:* There is a tuple $(1, x', y', z') \in \mathcal{T}$ such that $y \oplus z = y'$ or $y' \oplus z' = y$.

4. *Condition C_4:* There is a tuple $(2, x', y', z') \in \mathcal{T}$ such that $y \oplus z = x'$.

If the response y is such that at least one of these conditions is satisfied, then the simulator S_0 explicitly fails. We can estimate the occurrence probabilities for these failure condition similar to the case of a forward query $(1, +, x, y)$.

For queries made to the block cipher $E2$, we need to check for different failure conditions. In particular, the Merkle-Damgård construction using $E2$

will only be applied to one block inputs in the RO construction $\mathrm{NMAC}^{E1,E2}$. For forward queries $(2,+,x,y)$, the new simulator S_0 chooses $z \in \{0,1\}^{n'}$ in the same way as the original simulator S and sends z as its response without checking for any failure conditions. On the other hand, for inverse queries $(2,-,x,z)$, the simulator S_0 chooses $y \in \{0,1\}^{n'}$ similar to S, but then checks to see if the tuple (x,y,z) satisfies the following condition:

1. *Condition C'_1:* It is the case that $y = IV'$.

If the tuple (x,y,z) satisfies this condition and the response y was freshly chosen at random, then the simulator S_0 explicitly fails. The probability of occurrence of the failure condition C'_1 is a straightforward probability computation.

Let G_3 denote the event that the distinguisher D outputs 1 in game 3, i.e. $\Pr[G_3] = \Pr\left[D^{R_0^F, S_0^F}(1^\lambda) = 1\right]$. The response distribution of the distinguisher differs in games 2 and 3 if and only if the simulator S_0 fails in game 3. This event is identical to one of the failure conditions holding for the responses of the simulator S_0.

$$\begin{aligned} |\Pr[G_3] - \Pr[G_2]| &= \Pr[B_1 \vee B_2 \vee B_3 \vee B_4 \vee C_1 \vee C_2 \vee C_3 \vee C_4 \vee C'_1] \\ &\leq \frac{q^2}{2^{\min(n.n')}} \end{aligned}$$

Game 4. In this game, we modify the relay algorithm, but leave the ideal cipher simulator S_0 unchanged. In particular, the new relay algorithm R_1 does not simply relay the outputs of the random oracle F. Instead, R_1 is given oracle access to the simulator S_0, and it responds to any random oracle queries made to it by honestly evaluating the RO construction $\mathrm{NMAC}^{E1,E2}$ by using the simulator S_0 in place of the ideal ciphers $E1$ and $E2$.

Let G_4 denote the event that the distinguisher D outputs 1 in game 4, so that

$$\Pr[G_4] = \Pr\left[D^{R_1^{S_0^F}, S_0^F}(1^\lambda) = 1\right]$$

We assume that the maximum length of a random oracle query made by the distinguisher is $\ell \cdot \kappa$. Now we will show that the view of the distinguisher D does not change by a non-negligible amount when we make this change to the relay algorithm. This is formally stated below.

Claim 15. *Let G_3 and G_4 denote the events that the distinguisher outputs 1 in game 3 and 4, respectively. Let q_E and q_F denote the number of ideal cipher (including both $E1$ and $E2$ queries) and random oracle queries made by the distinguisher, then it is the case that*

$$|\Pr[G_4] - \Pr[G_3]| = \mathcal{O}\left(\frac{(q_E + q_F \cdot \ell)^2}{2^{\min(n,n')}}\right)$$

proof of claim 15: The view of the distinguisher changes in the transition from game 3 to 4 only if there is a change in the response distributions of either the relay algorithm or the simulator between the two games. We will show that if the simulator S_0 does not fail in either of the two games, then such a change in the response distributions is impossible.

Let us start by analyzing the way the two relay algorithms, R_0 and R_1, choose their responses. The relay algorithm from game 3, R_0, simply forwards the random oracle output to any RO query X (i.e. responds with $F(X)$). On the other hand, the relay algorithm from game 4 uses the block ciphers simulated by S_0 to implement the RO construction $\text{NMAC}^{E1,E2}$, and responds with the output of this "simulated construction". If the distinguisher detects a difference in the responses of the two relay algorithms, then it must be the case that the simulator S_0 did not adjust its responses consistently with the RO F in game 4, which resulted in the response of the relay algorithm R_1 not matching the RO output. We will show that unless the simulator S_0 explicitly fails, it is always able to adjust its responses consistent with the random oracle F.

The simulator S_0 is the same in both games 3 and 4. However, the simulator receives extra queries from the relay algorithm R_1 in game 4. Thus it may be the case that the simulator S_0 chooses its response to the same query differently, depending on whether it is in game 3 or game 4. This is the case only if the simulator chooses its response consistent with the RO F in one game, while independently at random in the other game. We will show that such a difference is impossible, unless the simulator S_0 explicitly fails in one of the games.

Below, for simplicity, we will denote by $\text{NMAC}^{S_0}(X)$ the output of the "simulated RO construction" $\text{NMAC}^{E1,E2}$ using the block ciphers simulated by S_0, while $F(X)$ is the actual random oracle output on X. We will start by proving a couple of useful properties of the responses of the simulator S_0

that hold in both games 3 and 4. The first property essentially says that if the simulator S_0 does not fail then it is not possible for the input to the Davies-Meyer function based on $E2$ to collide for two different RO inputs.

Claim 16. *If the simulator S_0 does not explicitly fail, then there are no two different sequences of κ-bit blocks $x_1 \dots x_m$ and $x'_1 \dots x_p$ with corresponding tuples $(1, x_1, y_1, z_1) \dots (1, x_m, y_m, z_m)$ and $(1, x'_1, y'_1, z'_1) \dots (1, x'_p, y'_p, z'_p)$ in the table $\mathcal{T}$ of S_0 such that:*

- *It is the case that $y_1 = y'_1 = IV$. Moreover, for each $b = 2 \dots m$ and $b' = 2 \dots p$, it holds that $y_b = y_{b-1} \oplus z_{b-1}$ and $y'_{b'} = y'_{b'-1} \oplus z'_{b'-1}$.*
- *It is the case that $y_m \oplus z_m = y'_p \oplus z'_p$.*

proof of claim 16: This is easy to see since there is $r \in \{0 \dots (\min(m,p) - 1)\}$ such that,

$$\begin{aligned} \forall s \in \{0, (r+1)\} \quad : \quad & (x_{m-s}, y_{m-s}, z_{m-s}) = (x'_{p-s}, y'_{p-s}, z'_{p-s}) \\ & \text{and } (x_{m-r}, y_{m-r}, z_{m-r}) \neq (x'_{p-r}, y'_{p-r}, z'_{p-r}) \end{aligned}$$

Of the two tuples $(1, x_{m-r}, y_{m-r}, z_{m-r})$ and $(1, x'_{p-r}, y'_{p-r}, z'_{p-r})$, we consider the one whose corresponding query was made later. Without loss of generality, let this be $(1, x_{m-r}, y_{m-r}, z_{m-r})$. If this was a result of a forward query $(1, +, x_{m-r}, y_{m-r})$, then the simulator S_0 would have failed due to failure condition B_2. On the other hand if this were an inverse query, then S_0 would have failed as a result of the failure condition C_2. $\square$

Next, we show that if the distinguisher wishes to find out the output $\mathrm{NMAC}^{S_0}(X)$ for a random oracle query $X = x_1 \parallel \ldots \parallel x_m$, then the only way it can do so is by computing the RO construction honestly.

Claim 17. *Consider any sequence of entries* $(1, x_1, y_1, z_1) \ldots (1, x_m, y_m, z_m)$ $, (2, x', y', z')$ *in the table* $\mathcal{T}$ *maintained by the simulator* S_0 *that satisfy the following properties:*

- *It is the case that* $y_1 = IV$ *and* $y' = IV'$*.*
- *For all* $i = 2 \ldots m$*, it is the case that* $y_i = y_{i-1} \oplus z_{i-1}$*.*
- *It also holds that* $x' = y_m \oplus z_m$*.*

If the simulator S_0 *does not explicitly fail, then it is necessarily the case that these entries were generated as a result of the ordered sequence of queries* $(1, +, x_1, y_1), \ldots, (1, +, x_m, y_m), (2, +, x', y')$*.*

proof of claim 17: To the contrary, assume that the tuples $(1, x_1, y_1, z_1) \ldots$ $(1, x_m, y_m, z_m), (2, x', y', z')$ were not generated as a result of the ordered sequence of forward queries $(1, +, x_1, y_1), \ldots, (1, +, x_m, y_m), (2, +, x', y')$. In this case, one of the following must hold:

1. The tuple $(1, x_m, y_m, z_m)$ was stored in the table $\mathcal{T}$ after the tuple $(2, x', y', z')$, as a result of a forward/inverse query.

2. For some $j \in \{1 \ldots (m-1)\}$, a new forward query $(1, +, x_j, y_j)$ was made when the tuple $(1, x_{j+1}, y_{j+1}, z_{j+1})$ already existed in the table $\mathcal{T}$.

3. For some $j \in \{2 \ldots m\}$, a new inverse query $(1, -, x_j, z_j)$ was made when the tuple $(1, x_{j-1}, y_{j-1}, z_{j-1})$ already existed in the table $\mathcal{T}$.

4. The tuple $(1, x_1, y_1, z_1)$ was stored in $\mathcal{T}$ as a result of an inverse query $(1, -, x_1, z_1)$.

5. The tuple $(2, x', y', z')$ was stored in $\mathcal{T}$ as a result of the inverse query $(2, -, x', z')$.

We will show how any of these situations would have resulted in the simulator S_0 explicitly failing. In each of these cases, we can deduce that at least one of the failure conditions would have held.

- Case 1 : In this case, the failure condition B_4 (resp. C_4) would have been true for the query $(1, +, x_m, y_m)$ (resp. $(1, -, x_m, z_m)$).

- Case 2 : Failure condition B_3 would have been true for the query $(1, +, x_j, y_j)$.

- Case 3 : Failure condition C_3 would have been true for the query $(1, -, x_j, z_j)$.

- Case 4 : Failure condition C_1 would have been true for the query $(1, -, x_1, z_1)$.

- Case 5 : Failure condition C'_1 would have been true for the query $(2, -, x', z')$.

Thus if the simulator never fails, then the sequence of tuples $(1, x_1, y_1, z_1) \ldots (1, x_m, y_m, z_m), (2, x', y', z')$ could have been stored only as a result of the sequence of forward queries $(1, +, x_1, y_1), \ldots, (1, +, x_m, y_m), (2, +, x', y')$. $\square$

As a consequence of claims 16 and 17, we can deduce that in both games 3 and 4 the simulator is always able to adjust its responses to be consistent with random oracle F if it does not explicitly fail. Thus the responses of the relay algorithm R_0 and R_1 are identical in the view of the distinguisher. Moreover, as a result of claim 17, we can also deduce that the distinguisher D can only find the output $\mathrm{NMAC}^{S_0}(X)$ by making the sequence of forward queries given in claim 17. In this case, the simulator adjusts its response accordingly so that $\mathrm{NMAC}^{S_0}(X) = F(X)$ for any X. Thus the view of the distinguisher D does not change in the transition between games 3 and 4 if the simulator S_0 does not explicitly fail in either game. Hence, we can deduce that

$$\begin{aligned} |\Pr[G_4] - \Pr[G_3]| &\leq \Pr[S_0 \text{ fails in either game}] \\ &= \mathcal{O}\left(\frac{(q_E + q_F \cdot \ell)^2}{2^{\min(n,n')}}\right) \end{aligned}$$

$\square$

GAME 5. In this game, we modify the simulator so that it always selects its responses independent of the random oracle F. This does not induce any inconsistencies in the view of the distinguisher since the relay algorithm R_1 also uses the new simulator S_1 instead of directly using the random oracle F.

The new simulator S_1 always chooses a uniformly random response to any query made to it, including any forward query $(2, +, x, IV')$. Moreover, after it chooses a response it does not check for any of the failure conditions that the old simulator S_0 checked for in game 4. The view of the distinguisher does not change by more than a negligible amount in the transition from game 4 to 5. This is because the distinguisher only notices a difference between the two games if S_0 fails in game 4 (or equivalently, the new simulator S_1 responds with a z that satisfies one of the failure conditions checked by S_0). Since the new simulator S_1 always chooses a uniformly random response to any query, we can easily bound this difference.

$$\begin{aligned}|\Pr[G_5] - \Pr[G_4]| &\leq \Pr[S_0 \text{ fails in game 4}] \\ &\quad + \Pr[S_1 \text{ satisfies one of the failure conditions}] \\ &= \mathcal{O}\left(\frac{(q \cdot \ell)^2}{2^{\min(n,n')}}\right)\end{aligned}$$

GAME 6. This is the final game of our proof. Here we replace the simulator S_1 by actual ideal block ciphers $E1 : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$ and $E2 : \{0,1\}^{\kappa'} \times \{0,1\}^{n'} \rightarrow \{0,1\}^{n'}$. Since the relay algorithm R_1 essentially implements the RO construction $\mathrm{NMAC}^{E1,E2}$, the view of the distinguisher in this game is identical to its view in the ideal cipher model.

Let G_6 denote the event that the distinguisher D outputs 1 in this game. We can deduce that the view of the distinguisher does not change in the transition from game 5 to 6, unless the simulator S_1 outputs a collision in block cipher outputs for the same key. The probability of this event can be bounded by simply using the birthday paradox.

$$\begin{aligned} |\Pr[G_6] - \Pr[G_5]| &\leq \Pr[S_1 \text{ outputs a collision.}] \\ &= \mathcal{O}\left(\frac{(q \cdot \ell)^2}{2^{\min(n,n')}}\right) \end{aligned}$$

Now we can complete the proof of lemma 3 by combining the above games. Hence, we deduce that

$$\left|\Pr\left[D^{\mathrm{NMAC}^{E1,E2},E1,E2}(1^\lambda) = 1\right] - \Pr\left[D^{F,S}(1^\lambda) = 1\right]\right| = \mathcal{O}\left(\frac{q^2\ell^2}{2^{\min(n,n')}}\right)$$

$\square$

Lemma 4. *The HMAC construction $HMAC^E$ using an ideal block cipher $E : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$ is (t_D, t_S, q, ϵ)-indifferentiable from a random* *oracle $F : \{0,1\}^* \rightarrow \{0,1\}^n$ in the ideal block cipher model for E, for any t_D and $t_S = \mathcal{O}(q^2)$, with $\epsilon = 2^{-n} \cdot \ell^2 \cdot \mathcal{O}(q^2)$ ($\ell\kappa$ is the maximum length of an RO query made by the distinguisher).*

Proof: The proof of this lemma is almost identical to the proof of indifferentiability for the NMAC construction given in lemma 3. This is because the HMAC construction essentially implements the NMAC using a single block cipher, by using different initialization vectors in each part of the construction. With slight modifications, the simulator described in lemma 3 works in this case as well.

The proof of indifferentiability is also almost identical to that in lemma 3. We do add a few extra "failure conditions" to handle the fact that we are using the same ideal cipher E in place of both E_1 and E_2. $\square$

3.2.4 Implications for the RO Domain Extenders

We saw above that the four modifications of the Merkle-Damgård construction, i.e. the prefix-free, chop, NMAC and HMAC constructions, applied to the Davies-Meyer compression function are indifferentiable from a variable-length input random oracle (VIL-RO) in the ideal cipher model. This fact was formally stated and proved in theorem 6. Now we will show that this result is stronger than the indifferentiability of domain extenders for the random oracle described in section 3.1. In particular, we show that theorems 2, 3, 4 and 5 from section 3.1 can be derived as a direct consequence of theorem 6.

To this purpose, say we are given a fixed-length input random function oracle (FIL-RO) $f : \{0,1\}^{\kappa+n} \to \{0,1\}^n$. Consider the following construction based on f:

$$\begin{aligned} T^f : \{0,1\}^\kappa \times \{0,1\}^n &\to \{0,1\}^n \\ (x, y) &\mapsto f(x \parallel y) \oplus y \end{aligned}$$

Note that the construction T^f is essentially the same as the Davies-Meyer construction except that the latter is defined for an ideal block cipher $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$. If we are able to show that T^f is indifferentiable from the ideal block cipher E, then it will complete the proof of all theorems from section 3.1 as an implication of theorem 6 and the composability property of indifferentiable constructions. This is because the Davies-Meyer construction applied to T^f is identical to the FIL-RO f. However, it is easily seen that T^f cannot be proven indifferentiable from the ideal cipher E[2].

To overcome this, we introduce a weaker ideal primitive than the ideal cipher, which we will call the *weak ideal block cipher*. A *weak ideal block cipher* E is essentially the same as an ideal cipher, except that it only responds to forward block cipher queries. In this case, we do not run into the problem of responding to inverse queries made to the construction T^f. Unfortunately, we

[2]In particular, the construction T^f cannot answer inverse ideal cipher queries.

cannot use theorem 6 in a "black-box manner" to get indifferentiable VIL-RO construction using a weak ideal cipher. However, none of the constructions proposed in theorem 6 make use of inverse queries to the underlying block cipher.

Corollary 1. *The block-cipher based constructions pf-*MD_g^E*, chop-*MD_s^E*,* $NMAC_g^E$ *and* $HMAC^E$ *are* (t_D, t_S, q, ϵ)*-indifferentiable from a random oracle, in the* weak ideal cipher model *for* E*, for any* t_D *and* $t_S = \ell \cdot \mathcal{O}(q^2)$*, with* $\epsilon = 2^{-n} \cdot \ell^2 \cdot \mathcal{O}(q^2)$ *for pf-*MD_g^E*,* $\epsilon = 2^{-s} \cdot \ell^2 \cdot \mathcal{O}(q^2)$ *for chop-*MD_s^E*,* $\epsilon = 2^{-\min(n,n')} \cdot \ell^2 \cdot \mathcal{O}(q^2)$ *for* $NMAC_g^E$ *and* $\epsilon = 2^{-\min(\kappa,n)} \cdot \ell^2 \cdot \mathcal{O}(q^2)$ *for* $HMAC^E$*. Here* ℓ *is the maximum message length queried by the distinguisher.*

In fact, the proof of this theorem is simpler than that for theorem 6 since the simulator need not respond to inverse ideal cipher queries. We now show that the construction T^f is an indifferentiable construction of a weak ideal cipher $E : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$ using the FIL-RO f.

Lemma 5. *The construction* T^f *(described above) is* (t_D, t_S, q, ϵ) *indifferentiable from a weak ideal cipher* $E : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$ *for any* t_D*,* $t_S = \mathcal{O}(q^2)$ *and* $\epsilon = 2^{-n} \cdot q^2$*, in the random oracle model for* f*.*

Proof: In order to prove this theorem, we need to describe a random oracle simulator S such that no distinguisher can tell apart the random oracle model, where it has oracle access to the random function oracle f and the construction T_f, from the weak ideal cipher model, where it has oracle access to the simulator S and the weak ideal block cipher E. We will start by describing the simulator.

The Simulator. The simulator S gets random oracle queries of the form $x \parallel y \in \{0,1\}^{\kappa+n}$. The simulator makes the forward query (x, y) to block cipher E to get $E_x(y)$. Then S responds with $z = E_x(y) \oplus y$. In addition, the simulator S also maintains a table $\mathcal{T}$ of previous query-response pairs $(x \parallel y, z)$ which it checks each time to see if the current query matches a previous one.

Proof of Indifferentiability. The proof of indifferentiability involves a hybrid argument that starts in the ideal cipher model, where the distinguisher D has oracle access to E and S, which is game 1.

Game 1. This is essentially the *weak ideal cipher model*, where the distinguisher D is given oracle access to the random oracle simulator S and the weak ideal cipher E. Let G_1 denote the event that D outputs 1 in this game. Thus, if λ denote the security parameter,

$$\Pr[G_1] = \Pr\left[D^{S^E,E}(1^\lambda) = 1\right]$$

Game 2. In this game, we give the distinguisher D oracle access to a relay algorithm R_0, instead of the weak ideal cipher E. This relay algorithm R_0 has oracle access to the simulator S^E. On a forward block cipher query $(x, y) \in \{0,1\}^\kappa \times \{0,1\}^n$, the relay algorithm R_0 simply queries the simulator

S_E on $x \parallel y$ to get its response z. Then R_0 responds to the block cipher query with $y \oplus z$.

Let G_2 denote the event that D outputs 1 in this game. Since the view of the distinguisher does not change in this game, we can deduce that

$$\Pr[G_1] = \Pr\left[D^{S^E, R^{S^E}}(1^\lambda) = 1\right] = \Pr[G_1]$$

Game 3. In this game, we modify the simulator so that it does not consult the ideal block cipher for any of the queries made to it. Instead, the new simulator S_0 always chooses a uniformly random n-bit response z to every new query $x \parallel y$, and records it in its table $\mathcal{T}$ before sending over the response.

Let G_3 denote the event that the distinguisher D outputs 1 in this game. Since the relay algorithm only consults S_0 for any query, so that the view of the distinguisher in this game is entirely independent of the weak ideal cipher E. Thus the distinguisher D detects a difference between this game and game 2 only if the relay algorithm R_0 outputs a collision for two block cipher queries with the same key, and the probability of this event can be easily bounded using the birthday paradox. Thus, we can deduce that

$$|\Pr[G_3] - \Pr[G_2]| \leq \frac{q^2}{2^n}$$

Note that the simulator S_0 is essentially the same as the fixed-length input RO f, while the relay algorithm R_0 is defined in the same way as the construction T^f. Hence, we can also deduce that

$$\begin{aligned}\left|\Pr\left[D^{f,T^f}(1^\lambda)=1\right]\Pr\left[D^{S^E,E}(1^\lambda)=1\right]\right| &= |\Pr[G_3]-\Pr[G_1]| \\ &\le \frac{q^2}{2^n}\end{aligned}$$

$\square$

3.3 Other Extensions

Increasing Output Length. All the random oracle constructions that we have discussed, permit really efficient output expansion. Given a random oracle $H : \{0,1\}^* \to \{0,1\}^n$, output expansion by a factor L can be achieved by appending an extra $\log(L)$-bit block to the input X and outputting the concatenation of the following blocks:

$$H(X \parallel \langle 1\rangle), H(X \parallel \langle 2\rangle), \ldots, H(X \parallel \langle L\rangle)$$

It can be easily seen that this construction is generically secure, including any of the indifferentiable constructions of VIL-RO that we have proposed. However, one would imagine that evaluating this construction would involve L evaluations of the VIL-RO H.

As it turns out, for the Prefix-free, Chop, NMAC and HMAC constructions of a VIL-RO using a FIL-RO or an ideal cipher, this procedure can be completed extremely efficiently using only one (or two) extra evaluation of the underlying fixed-length input primitive for each extra block of output. [3] This can be done by first computing the Merkle-Damgård construction on the input X, and evaluating only one last part of the construction for each of the output blocks. This reduces the running time for the procedure from $L \cdot (|X|/\kappa)$ to $L + (|X|/k)$ computations.

Domain Separation for Independent ROs. The same technique as above can also be used for domain separation of the random oracle, to get multiple independent random function oracles from a single one. This is useful in cryptographic constructions where one needs to use multiple independent random oracles in order to prove the security of the construction. In particular, if we have a single random oracle $H : \{0,1\}^* \rightarrow \{0,1\}^n$, and we need L independent random oracles in our constructions, then we can achieve this by defining these random oracles as:

$$\begin{array}{rcl} H_1(X) & := & H(X \parallel \langle 1 \rangle) \\ & \vdots & \\ H_L(X) & := & H(X \parallel \langle L \rangle) \end{array}$$

We cannot use the same efficient processing technique that we used for output expansion, since one usually does not need to evaluate the independent

[3]For a prefix-free encoding g, this can be done by appending $\langle 1 \rangle \ldots \langle L \rangle$ to $g(X)$ instead appending to X and then evaluating g.

random oracles on the same input.

Chapter 4

Getting the Best out of existing Hash Functions

In the previous chapter, we discussed the security of hash functions when used to instantiate the random oracle. This was a really strong security requirement from hash functions and, not surprisingly, one needs to make ideal assumptions on the compression function to prove the security of the iterative hash function in this case. Here we will take a more general look at iterative hash functions without restricting to some particular security requirement.

As we have already seen, *cascade chaining* is a very elegant way to build a hash function H on arbitrary-length inputs from a given compression function h on fixed-length inputs. Recall that for a given $h : \{0,1\}^{\kappa} \times \{0,1\}^{n} \to \{0,1\}^{n}$, one can define a hash function H, parametrized by an initialization vector $IV \in \{0,1\}^n$, as follows (where input $x = x_1 \parallel \ldots \parallel x_\ell$ and $x_i \in \{0,1\}^\kappa$ for $i = 1 \ldots \ell$):

$$H(x_1 \parallel \ldots \parallel x_\ell) = h(x_\ell, h(\ldots, h(x_1, IV) \ldots))$$

We will refer to this as the MD mode (after Merkle-Damgård). The most abundant use of the MD mode in practice comes in the design of the industry-standard hash family SHA (which consists of several specific hash functions SHA-x, where $x \in \{1, 224, 256, 384, 512\}$). Unfortunately, despite its elegance and simplicity, the "plain MD" mode has several deficiencies. For instance, it does not guarantee that a "global" collision of H implies a "local" collision of the compression function h, unless one preprocesses the input into a suffix-free form before applying H [22] (as we already mentioned, the particular suffix-free encoding of appending the message length is called *MD strengthening*, and is actually used in the SHA family for this reason). More seriously, as we already saw in chapter 3, even MD strengthening falls prey to the "extension attack" [1] which makes it insufficient for domain extension of random oracle. Moreover, this deficiency disqualifies the natural use of "plain MD" in the design of "pseudorandom functions" [5]. Other problems also arise when the MD mode is used in applications such as key derivation [25] and target collision-resistance (or UOWHFs [2]) [11, 70].

Apart from the issues mentioned above, several other deficiencies of the

[1]given $H(x)$ and any extension y, one can compute $H(x \parallel y)$ without knowing x.
[2]Universal One-Way Hash Functions

MD mode against exponential-time attacks have been discovered [43, 45]. All these deficiencies, coupled with the improved brute-force attacks on the popular SHA-1 hash function proposed recently [72, 73], suggest that it is time to design a better, more "secure" mode of operation for building a variable-length input hash function. With this purpose, NIST has been organizing several workshops dedicated to coming up with the next generation hash functions [62]. However, this process will take some time, and it does not appear that such hash functions would be standardized and widely accepted in any forseeable future. Therefore, practitioners are "stuck" with the prospect of using existing hash functions, despite all their deficiencies. Hence, there is a pressing need to design immediate "fixes" to the MD paradigm, without changing it drastically.

There are two aims in coming up with such "fixes" to the MD mode. The first, and so far the most popular, aim is to design a slight variant of the MD mode that provably preserves a given security property of the compression function, and to do so in the most aesthetic and efficient manner. We mention only a few of the many examples of this approach. For collision-resistance, we already mentioned the well known technique of *MD strengthening*. For another example, by viewing the initialization vector as the key and applying a *prefix-free encoding* to the message, one can obtain a variable-length input pseudorandom function from a fixed-length input pseudorandom compression function [5]. In the case of target collision-resistance, Shoup [70] designed an elegant mode for building target collision-resistant (TCR) hash functions (or UOWHFs [60]) from a TCR compression function by cleverly XORing certain masks to the internal chaining variables in the MD construction. The common feature in all these results is that one assumes *exactly the same* property from the compression function h as the desired property from the hash function H. In many cases, such as the PRF and TCR examples, this means that a "secure" mode must be sufficiently different from the plain MD so that its implementation requires a non-trivial modification to the SHA implementation. Concretely, the SHA family uses a fixed public IV (as opposed to arbitrary secret IV needed for PRFs), while in the TCR case one cannot XOR the corresponding masks without modifying the internals of SHA.

The second, less popular, aim is to try and design a "secure" mode that uses only black-box calls to the plain MD mode [3]. For instance, MD strengthening satisfies this property. Other examples include the HMAC mode for pseudorandom functions [5] and the results for domain extension of random oracle mentioned in the previous chapter. The attractive feature of these results is that they result in a hash function with the desired property without tinkering with the internals of SHA, and can use any off-the-shelf implementation. Moreover, all these examples also satisfy the *property-preserving* property described above.

Our Goal. In this chapter, we will emphasize the latter aim in coming up with "fixes" for existing hash functions. That is, we consider the question

[3]in practice, with MD strengthening, but we ignore this aspect for now

of building a hash function H' achieving a given security property P using a black-box MD-based hash function H (with an unknown compression function h). We require that the proposed construction H' satisfies the following "axioms":

1. The construction should consist of one or two *"black-box" calls* to H. In particular, the construction is not allowed to use any knowledge of or tinker with the internals of the hash function H.

2. The construction must support variable-length inputs.

3. Compared to a single evaluation of $H(M)$, the evaluation of $H'(M)$ should make at most a fixed (small constant) number of extra calls to the underlying compression function of H. In other words, the efficiency of H' is negligibly close to that of H.

The motivation behind requiring the construction H' to satisfy these axioms is from the viewpoint of a practitioner who understands the properties of the hash function that are needed for the security of his cryptosystem, but who wants to use an off-the-shelf standardized hash function implementation without tinkering with its internals. Such a practitioner would be willing to sacrifice the *property-preserving* aspect of the "fix" in favor of a black-box implementation.

In fact, the above "axioms" leave very little freedom in choosing the modes of operation for H'. The resulting modes are essentially the *most widely-utilized* constructions appearing in practical implementations:

1. *Plain MD Construction:* This captures the notion that the application uses the hash function as it is. We will denote this mode of operation as H.

2. *Encode-then-MD Construction:* In this case, the user encodes the hash function input before applying the plain MD construction. Examples of popular encoding schemes used are suffix-free encoding and prefix-free encoding. We will refer to the corresponding constructions as the *prefix-free MD construction* $\mathcal{H}_{pre}$ and the *suffix-free MD construction* H_{suf}.

3. *MD-then-Chop Construction:* Here the user applies the plain MD mode and only uses part of the output while discarding the remaining bits. In particular, existing hash functions SHA-224 and SHA-384 are obtained this way from SHA-256 and SHA-512, respectively. We denote the MD-then-chop construction that chops s bits of the output as H_{chop_s}.

4. *NMAC/HMAC Construction:* The version of the NMAC construction that we consider simply composes two applications of the plain MD mode with possibly different initialization vectors IV_1 and IV_2. While not obeying the first axiom, the NMAC construction serves as a nice abstraction for the HMAC construction which does satisfy all our axioms (but is slightly harder to formally analyze in some cases). Concretely, the HMAC construction uses the NMAC construction with $IV_1 = h(IV, \alpha_1) = H(\alpha_1)$ and $IV_2 = h(IV, \alpha_2) = H(\alpha_2)$, where each

α_i is either the null string $\perp$ (in which case we let $h(IV, \perp) = IV$) or a single κ-bit block. We denote the NMAC construction as H_{nmac} and the HMAC construction as H_{hmac}.

Now we can finally rephrase our goal as follows. Given a particular desired security property P (such as collision-resistance or pseudorandomness) and one of the 4 modes of operation above (which all satisfy our axioms), find the weakest security assumption(s) P' on the compression function h which would make the corresponding mode satisfy P (or determine that the construction is insecure for any h). Ideally, this security property P' for h would be P itself (which would result in a *property-preserving mode of operation*). However, unlike most previous work, property preservation is not our primary concern. In particular, we will not declare a mode of operation to be "insecure" for a property P simply because it is not property-preserving for P. Instead, we will find the weakest security property P' of the compression function that makes the resulting construction secure. This will allow the practitioners to decide whether or not it is reasonable to assume that the compression function of existing hash functions, such as SHA, satisfy the property P'.

Our Results. We achieve our main goal for a very wide variety of security properties including *collision-resistance (CR)*, *pseudorandomness (PR)*, *indifferentiability from random oracle (RO)*, *message authentication (MAC)*, *target collision-resistance (TCR)*, *second preimage-resistance (SPR)*, *randomness extraction (RE)* and *one-wayness (OW)*. In each case, and for each of the four popular modes above, we will identify the needed property P' on

h. In some cases, the needed P' easily follows from some existing work (for instance, from the previous chapter or [21] in the case of domain extension of random oracle). In other cases, it required some minor, but important modifications to the existing results in order to satisfy our axioms. For example, by assuming that "$h(IV, random) = random$" in addition to h being a PRF when keyed with the first n bits of its input, we could build a variable length PRF using the encode-then-MD mode and adjusting the proof of [5]. More interestingly, by making extra assumptions on h, in some cases we can prove security of the modes which were previously believed "insecure" because they were not property-preserving. Finally, in some cases the proof will involve careful and non-trivial modification of previous results. For example, this is the case when analyzing the one-wayness of the H_{suf} construction.

In addition to giving an exhaustive "mode $\times$ property" guide (see table 4.1) for achieving a given security property with a given popular mode, in each section we also mention the practical implication of our results when using existing hash functions SHA-x, where $x \in \{1, 224, 256, 384, 512\}$.

Related Work. We have already cited many of the relevant papers. In particular, the variants of the MD mode that are useful in the property-preservation of collision-resistance [22], pseudorandomness [5, 6], message-authentication [1, 53], random oracles [21] and randomness extraction [25]. We also mention the works of [12, 13] concerned with multiple property-preservation; namely, designing a single mode of operation which simultaneously preserves several properties. Unfortunately, the modes of [12, 13] do

	Plain MD	Encode-then-MD	MD-then-Chop	NMAC/HMAC
CRHF	(1) + (2)	Suf-Free+(1) Pre-Free+(1)+(2)	(1') + (2)	N/HMAC+(1)+(2) $\alpha_1 \neq \perp$
PRF	Append key + (1)+(2)+(4)	SF+(1)+(4) (append) PF+(2')+(3) (prepend)	Prepend key + (2')+(3')	N/H+(3)+(4) (prepend) Any $IVs/\alpha s$
RO	Not Secure	Suf-Free not secure Pre-Free+(5)	(5) worse security	NMAC/HMAC+(5) $IV_1 \neq IV_2$; $\alpha_1 \neq \alpha_2$
MAC	Append key + (1)+(2)+(6)	SF+(1)+(6) (append) PF+(1)+(2)+(6)(app.)	Append key + (1)+(2)+(6')	N/H+(1)+(2)+(6) Any IVs/αs
TCR	$key \oplus blks$ (7) + (9)	SF+(7) ($key \oplus blks$) PF+(7)+(9)	$key \oplus blks$ (7') + (9)	N/H+(7)+(9) (append) Any IVs/αs ($key \oplus blks$)
SPR	(8) + (9)	SF+(9) PF+(8)+(9)	(8') + (9)	N/H+(8)+(9) Any IVs/αs
RExt	(10) $H_\infty(M) \wedge H_\infty(m_\ell)$	MDS + (10)(SF/PF??) $H_\infty(M) \wedge H_\infty(m_\ell)$	(10) $H_\infty(M)$	NMAC + (10) HMAC??
OWF	(2)+(11)	MDS+(2)+(11) (SF/PF??)	(2')+(11)	NMAC+(2)+(11) HMAC??

Assumptions on compression function:

(1)=Collision Resistance (CR)	(1')=CR after Chop
(2)=Output Regular	(2')=$h(U_n, \cdot)$ is output regular
(3)=standard PRF (sPRF)	(3')=sPRF after Chop
(4)=dual PRF (dPRF)	
(5)=FIL-RO	
(6)=MAC with κ-bit key	
(7)=enhanced SPR (eSPR)	(7')=eSPR after Chop
(8)=computed SPR (cSPR)	(8')=cSPR after Chop
(9)=Fixed-point at random IV	
(10)=Family of random functions	
(11)=One-way function	

Misc.

SF=Suffix-free
PF=Prefix-free
MDS=MD Strengtheining
??=not known to be secure
RExt=Randomness Extrn.
$Key \oplus Blks$ =XOR key to each block

Figure 4.1: Table for comparing Security Property vs. Mode of operation.

not satisfy our axioms. Finally, we mention the work of Halevi and Krawczyk [40], which concentrated on building TCR hash functions, and is the closest in spirit to our motivation (indeed, we will use their results when discussing the TCR property). The authors built TCR hash functions using the encode-then-MD mode, and showed a simple coding scheme that yields a secure TCR hash function under an appropriately strong assumption on the underlying compression function h (still weaker than CR, but stronger than TCR).

Location of the key in keyed constructions. We note that for keyed constructions, such as constructions of pseudorandom and TCR functions, there are more than one possibilities for each hash function mode of operation. In particular, any construction for these primitives must specify the location of the key. In keeping with the black-box nature of the modes of operation, we prevent popular keying methods such as setting the key to be the IV or XORing the key into the chaining variables since this violates our basic axioms.

Moreover, we also do not consider the dedicated-key setting [1, 13], where there is separate space for the key in each application of the compression function. This is because existing hash functions do not support such dedicated keys. Even though we may consider the key to be part of the message block bits, we do not analyze this method since it yields constructions with poor input bandwidth (thus violating our last axiom). Hence, we will only consider modes of operation which incur an additive constant overhead compared to the plain MD mode.

4.1 Preliminaries

In this chapter, we will be interested more in the qualitative aspects of the security of iterative hash functions rather than focusing on the exact security in each case. For this purpose, we will give here slightly "less formal" and asymptotic definitions for each of these security notions related to hash functions. In particular, we will redefine some of the security notions already defined in chapter 2 (where these definitions were in "exact security" terms).

4.1.1 Collision Resistance

In this chapter, a collision resistant function ensemble H_λ is defined for a sequence of sets $\left\{\{0,1\}^{m(\lambda)}, \{0,1\}^{n(\lambda)}\right\}_{\lambda \in \mathbb{N}}$, where m and n denote the input and output length of H_λ, respectively. As in chapter 2, it consists of a pair of PPT machines $(Gen, Eval)$. However, we will give an asymptotic version of the definition of collision resistance here.

Definition 10. *ϵ-CR function family A function ensemble H_λ is a ϵ-collision resistant function family if for any probabilistic polynomial time machine A:*

$$\Pr\left[h_s(x_1) = h_s(x_2) \,\middle|\, s \leftarrow Gen(1^\lambda);\ (x_1, x_2) \leftarrow A(1^\lambda, s)\right] \leq \epsilon$$

Here ϵ is a function of the security parameter λ.

4.1.2 Pseudorandomness

Here a pseudorandom function ensemble H_λ is defined for a sequence of sets $\left\{\{0,1\}^{m(\lambda)}, \{0,1\}^{n(\lambda)}\right\}_{\lambda\in\mathbb{N}}$. It consists of a pair of PPT machines $(Gen, Eval)$, the key generation and evaluation machines.

Definition 11. *ϵ-PRF family Let R_λ be the truly random function ensemble. A function ensemble H_λ is a ϵ-pseudorandom function family if for any PPT oracle machine A:*

$$\left|\Pr\left[A^{h_s}(1^\lambda) = 1 \,\middle|\, s \leftarrow Gen(1^\lambda)\right] - \Pr\left[A^f = 1 \,|\, f \leftarrow R_\lambda\right]\right| \leq \epsilon$$

Here ϵ is a function of the security parameter λ.

4.1.3 Unpredictability and MACs

A message authentication code, MAC, is defined for a sequence of sets $\{\mathcal{M}_\lambda, \mathcal{T}_\lambda\}_{\lambda\in\mathbb{N}}$. It consists of a triple (Gen, Tag, Ver) of PPT machines, denoting the key generation, tagging and tag verification algorithms.

Definition 12 (ϵ-secure MAC). *A MAC (Gen, Tag, Ver) is a ϵ-secure MAC if for any PPT oracle machine A that outputs a message/tag pair (m, t) such that it never queried the tagging oracle on the message m:*

$$\Pr\left[Ver_s(m,t) = accept \,\middle|\, s \leftarrow Gen(1^\lambda);\ (m,t) \leftarrow A^{Tag_s, Ver_s}(1^\lambda)\right] \leq \epsilon$$

Here ϵ is a function of the security parameter λ.

4.1.4 Target Collision Resistance and One-Wayness

Target collision resistance is a weaker notion of collision intractability that collision resistance. A target collision resistant function ensemble is also called a *Universal One-Way Hash Function* ensemble (or simply UOWHFs). A TCR function ensemble is defined for a sequence of sets $\left\{\{0,1\}^{m(\lambda)}, \{0,1\}^{n(\lambda)}\right\}_{\lambda \in \mathbb{N}}$, and consists of a pair of algorithms $(Gen, Eval)$. However, the TCR attacker is more restricted than the collision finding attacker above, since it chooses one of the colliding inputs without knowledge of the hash function key.

Definition 13 (ϵ-TCR function family). *A function ensemble H_λ is a ϵ-secure TCR function family if for any pair of PPT machines (A_1, A_2):*

$$\Pr\left[h_s(x_1) = h_s(x_2) \,\middle|\, (x_1, \alpha) \leftarrow A_1(1^\lambda);\ s \leftarrow Gen(1^\lambda);\ x_2 \leftarrow A_2(1^\lambda, \alpha, x_1, s)\right] \leq \epsilon$$

Here ϵ is a function of the security parameter λ.

A notion related to TCR hash functions is that of *second preimage-resistant functions.* Unlike TCR hash functions this security notion is related to unkeyed hash functions $f : \{0,1\}^m \to \{0,1\}^n$ (where we can think of m as being the security parameter).

Definition 14 (ϵ-SPR function). *A function $f : \{0,1\}^m \to \{0,1\}^n$ is* ϵ-second preimage resistant *if for any PPT machine A:*

$$\Pr\left[f(x) = f(x') \,\middle|\, x \xleftarrow{\$} \{0,1\}^m;\ x' \leftarrow A(1^\lambda, x)\right] \leq \epsilon$$

Related to the notion of SPR functions, we can also define the notion of *preimage resistance* or *one-wayness*. This is a slightly weaker property than second preimage resistance.

Definition 15. *ϵ-secure one way function A function $f : \{0,1\}^m \rightarrow \{0,1\}^n$ is an* ϵ-secure one way function *if for any PPT machine A:*

$$\Pr\left[f(x) = y \middle| y \xleftarrow{\$} \{0,1\}^n;\ x \leftarrow A(1^\lambda, y)\right] \leq \epsilon$$

4.1.5 Randomness Extraction

A *randomness extractor* is a function that is used to extract uniformly random bits from inputs samples from an imperfect source of randomness. This has been an extremely useful primitive in cryptography, as well as theoretical computer science in general. We will give here brief definitions for this primitive.

We start by defining the notion of *min entropy*, which is a measure of the amount of randomness in a probability distribution. For instance consider a distribution $\mathcal{X}$ over $\{0,1\}^n$. The *min entropy* of the distribution $\mathcal{X}$, denoted as $H_\infty(\mathcal{X})$, is the minimum integer m such that $\Pr_{\mathcal{X}}(x) \leq 2^{-m}$ for all $x \in \{0,1\}^n$. Here $\Pr_{\mathcal{X}}(x)$ denotes the probability assigned to x by the distribution $\mathcal{X}$.

We will also need a way to quantify the distance between two probability distributions, $\mathcal{X}_1$ and $\mathcal{X}_2$, over a set S. The popular measure in this case is *statistical distance* between $\mathcal{X}_1$ and $\mathcal{X}_2$. The *statistical distance* between $\mathcal{X}_1$ and $\mathcal{X}_2$ is defined as $\mathbf{SD}(\mathcal{X}_1, \mathcal{X}_2) \overset{def}{=} \frac{1}{2}\sum_{s \in S} |\Pr_{\mathcal{X}_1}(x) - \Pr_{\mathcal{X}_2}(x)|$. If two distributions have statistical distance ϵ between them, then they are called ϵ-close distributions.

A *randomness extractor* is a function $h : \{0,1\}^\kappa \times \{0,1\}^m \to \{0,1\}^n$ that takes a κ-bit uniformly random seed and a m-bit input, and outputs a n-bit output.

Definition 16 ((k, ϵ) Extractor). *A (k, ϵ) extractor is a function $f : \{0,1\}^\kappa \times \{0,1\}^m \to \{0,1\}^n$ such that for every distribution $\mathcal{X}$ on $\{0,1\}^\kappa$ with $H_\infty(\mathcal{X}) \geq k$, the distribution $f(\mathcal{X}, U_m)$ is ϵ-close to the uniform distribution on $\{0,1\}^n$, where U_m denotes the uniform distribution on $\{0,1\}^m$.*

4.2 Security of MD modes

4.2.1 Collision Resistance

We will analyze each of the four modes described above for the minimal assumptions required on the compression function $h : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ needed in order to prove its collision resistance. As we discussed, we will not restrict ourselves to the case of *property preservation.* In particular, the security property needed for the compression function h may be stronger than collision resistance.

Plain Merkle-Damgård construction

It is a well-known fact that simply assuming collision resistance of the compression function does not suffice to prove collision resistance of the plain MD construction. Indeed, if the compression function h has a *fixed-point* such that there is some $x \in \{0,1\}^\kappa$ such that: $h(x, IV) = IV$. Then the output of the plain MD construction H collides for the inputs x and $x \parallel m$, for any m. Fortunately, if the compression function does not have any such fixed point then the plain MD construction H can be shown to be collision resistant.

We will state the following lemma in terms of simple security conditions on the compression function h. In the process, we introduce a new security property that essentially implies that the compression function is a *regular function.*

Assumption 1 (Regularity of outputs). *A function $h : \{0,1\}^m \to \{0,1\}^n$ is a ϵ* output regular function *if for any efficient machine A that gives a* 1 *bit output:*

$$|\Pr[A(x) = 1 \,|x \leftarrow h(U_m)] - \Pr[A(x) = 1 \,|x \leftarrow U_n]| \leq \epsilon$$

Here U_m and U_n denote the uniform distributions on $\{0,1\}^m$ and $\{0,1\}^n$, respectively.

Now we state the conditions required in order for the plain MD construction H to be collision resistant.

Lemma 6. *The plain MD construction* H *using a compression function* $h : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$ *is a* $\mathcal{O}(\ell \cdot (\epsilon_{reg} + \epsilon_{col}))$ *collision resistant hash function* [4] *if and only if* h *satisfies the following properties:*

- h *is* ϵ_{col} *collision resistant.*
- h *is an* ϵ_{reg} *output regular function.*

Proof: The main idea in the proof is to show that output regularity implies that no efficient attacker can find a fixed point in the compression function $h : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$ with non-negligible probability. That is, there is a negligible ϵ such that for all efficient attackers A:

$$\Pr\left[h(x_i, h(\ldots, h(x_1, IV)\ldots)) = IV \,\middle|\, IV \overset{\$}{\leftarrow} \{0,1\}^n;\ x_1 \ldots x_i \leftarrow A(IV)\right] \leq \epsilon$$

To the contrary, say there is an efficient attacker that finds such a fixed point with non-negligible probability ϵ', then we can show that it either breaks the collision resistance or the output regularity assumption for the compression function.

In order to show this, choose the initialization vector IV as $IV \leftarrow h(x)$ (for $x \leftarrow U_\kappa \times U_n$), instead of $IV \leftarrow U_n$. If the success probability of A changes by a non-negligible amount then we can break the output regularity assumption. Otherwise, the attacker A still finds, with non-negligible probability, a sequence of κ-bit blocks $x_1 \ldots x_i$ such that $h(x_i, h(\ldots, h(x_1, IV)\ldots)) =$

[4] ℓ denotes the maximum number of κ-bit blocks throughout this section

IV. Since A is unlikely to guess the preimage x of IV, it is likely to find a collision for h. Thus maximum success probability of an efficient attacker in finding such a fixed point is $\epsilon_{reg} + \epsilon_{col}$.

Now that we have shown that no efficient attacker is likely to find fixed points in h, we can essentially use the original proof of Merkle-Damgård [22, 54] to show that the plain MD construction H is collision resistant as well. $\square$

Encode-then-MD construction

It makes sense to only consider deterministic input coding schemes, since the resulting construction must behave like a function. We analyze two of the most popular such coding schemes, i.e. *prefix-free encoding* and *suffix-free encoding*.

We first not that using a prefix-free encoding on the input does not enable us to get rid of any security properties in lemma 6. Hence we can essentially restate the same result for the prefix-free MD construction H_{pre} as well. On the other hand, if we use a *suffix-free encoding* (such as Merkle-Damgård strengthening) then the resulting suffix-free MD construction H_{suf} can be shown to be collision resistant by simply assuming the collision-resistance of the compression function h [22, 54].

MD-then-Chop construction

Note that simply assuming collision resistance of the compression function is not useful for this construction, since we truncate s bits of the output. For instance, consider the case when h is collision resistant on these s bits, while is the constant function for all other bits (noted by Kelsey [44]). However, in our setting this only means that we need to make a stronger assumption on the compression function h. In particular, we will instead assume that h is collision resistant even if we remove these s bits from its output.

Lemma 7. *The MD-then-chop construction H_{chop_s}, using a compression function $h : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$, is a $\mathcal{O}(\ell \cdot (\epsilon_{reg} + \epsilon'_{col}))$ collision resistant hash function if the following holds:*

- *The function $h' : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^{n-s}$ defined as $h'(x,y) = h(x,y)|_{n-s}$ (i.e. chopping the last s bits from the output of h) is a ϵ'_{col} collision resistant function.*
- *h is a ϵ_{reg} output regular function.*

The proof of this lemma is essentially the same as for lemma 6.

NMAC/HMAC construction

We note that using the NMAC construction H_{nmac} does not help in improving upon the collision resistance of the plain MD construction H. This is essentially because any collision in the first the plain MD construction of H_{nmac}

(using initialization vector IV_1) essentially implies a collision for the entire construction. Hence, at best, we can restate lemma 6 for this construction as well.

Since the HMAC construction H_{hmac} is simply a black-box instantiation of the NMAC construction, this does not help in improving the collision resistance as well. However, we note that this construction has the best exact security if $\alpha_1 \neq \perp$.

4.2.2 Pseudorandomness

An issue in the pseudorandomness analysis of the MD modes of operation is the location of the PRF key. As discussed above, we need to specify the location of the key such that the resulting construction is still a black-box variant of plain MD. For our analysis, we will assume the key length to be the length of a single block (i.e. κ bits for the compression function $h : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$), and we will denote the key as K. We will analyze two approaches for keying each MD mode of operation:

1. *Prepend the key to input:* The PRF construction H outputs $H(K \parallel X)$ on input X.

2. *Append the key to input:* The PRF construction H outputs $H(X \parallel K)$ on input X.

Moreover, we will need two versions of pseudorandomness definitions for the compression function, one where the key occupies the last n bits and other where it occupies the first κ bits. We get the following two assumptions on the compression function in this manner.

- *Standard PRF (sPRF) security:* Here we require that for a uniformly chosen $K \in \{0,1\}^n$, the function $h(\cdot, K)$ must be indistinguishable from a truly random function.
- *Dual PRF (dPRF) security:* Here we require that for a uniformly chosen $K \in \{0,1\}^\kappa$, the function $h(K, \cdot)$ must be indistinguishable from a truly random function.

Depending on the maximum distinguishing advantage ϵ of an efficient attacker in each case, we call the compression function h ϵ-sPRF or ϵ-dPRF.

Plain MD construction.

In this case if we prepend the PRF key to the hash function input, then the resulting construction is not a PRF. This is because an attacker can use the *extension attack* to find $H(K \parallel X \parallel Y)$ by simply knowing the output $H(K \parallel X)$ and computing the compression function on the remaining blocks itself (where it does not need to know the key K). On the other hand, if we append the PRF key to the input, then we can show that if the plain MD construction using h is collision-resistant and satisfies the dual PRF security, then the plain MD construction $\mathsf{H}(\cdot \parallel K)$ is a variable-length input PRF.

Lemma 8. *The plain MD construction* H *is a* $\mathcal{O}(\ell \cdot (\epsilon_{col} + \epsilon_{reg}) + \epsilon_{dprf})$ *PRF (with PRF key appended to the function input) if the following conditions* *hold:*

- *h is ϵ_{col} collision resistant.*
- *h is a ϵ_{reg} output regular function.*
- *h is a ϵ_{dprf} dual pseudorandom function.*

Proof: If the PRF attacker cannot find any collisions in the plain MD construction H, then the n bit chaining variable [5] is unique for each PRF input. In this case, the dual PRF security of h implies the PRF security of the entire construction (with the same advantage). On the other hand, the attacker can find a collision in H with probability at most $\mathcal{O}(\ell \cdot (\epsilon_{reg} + \epsilon_{col}))$. $\boxminus$

Encode-the-MD construction.

Once again, we will discuss two deterministic coding schemes here, *prefix-free encoding* and *suffix-free encoding*. Let us first analyze the suffix-free MD construction H_{suf}. If we prepend the key to the (encoded) input, the resulting construction is still insecure since the *extension attack* works in this case as well. On the other hand, if we append the key to the (encoded) input then the resulting construction is a PRF if the suffix-free MD construction

[5]The chaining variable denotes the n bit intermediate inputs/outputs in the MD construction

H_{suf} using the compression function h is a dual PRF and collision resistant (for which we only need collision resistance of h in this case).

For the prefix-free MD construction H_{pre}, if we append the key to the (encoded) input then we get no advantage as compared to the plain MD construction and we can only restate lemma 8 in this case. On the other hand, if we prepend the PRF key to the (encoded) input then the resulting construction is not vulnerable to the *extension attack* in this case. Indeed, it was shown by Bellare et al. in [5] that the prefix-free MD construction with the PRF key in the IV is a PRF only assuming that the compression function h satisfies the standard PRF security. However, since we will need to prepend the key to the input (in order to preserve the black-box property of the construction), we will need to impose an extra condition on the compression function. In particular, we require that the function defined as $h(U_n, \cdot)$ is an output regular function. That is, if the first n bits of the compression function h are chosen at random then the resulting function is output regular with high probability.

Lemma 9. *The prefix-free MD construction* H_{pre} *is a* $\mathcal{O}(\epsilon'_{reg} + \ell \cdot \epsilon_{sprf})$ *secure PRF (with PRF key prepended to the input) if the following conditions hold:*

- h *is a* ϵ_{sprf} *sPRF.*
- $h(U_n, \cdot)$ *is a* ϵ'_{reg} *output regular function.*

Proof: The proof of this lemma essentially follows from the result of [5]. Indeed, if the attacker succeeds with non-negligible probability when the PRF key is prepended to the encoded input while has at most negligible success probability if the PRF key is in the IV of the construction, then it violates the output regularity property of the compression function. This is because for a random and secret key as input to $h(U_n, \cdot)$, the output is random and secret as well (if the output regularity property holds). $\sqcup\!\!\!\!\sqcap$

MD-then-Chop construction.

If the PRF key is appended to the input to the MD-then-Chop construction H_{chop_s}, then a slight variant of lemma 8 can be stated for this construction as well. Indeed, all we need is to specify the dual PRF and collision-resistance properties for the compression function with chopped output.

On the other hand, if we prepend the PRF key to the input to H_{chop_s}, then the extension attack does not seem to go through as in the case of plain MD construction. This is because the attacker does not learn the chopped s bits of the chaining variable by observing the output of H_{chop_s} for the prefix of an input. Indeed, this construction can be proven to be an arbitrary-length input PRF by making a slightly non-standard assumption on the compression function. In particular, we require the compression function to satisfy the following *resilient sPRF* assumption:

Assumption 2 ((s, ϵ)-resilient sPRF). *The function $h : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ is a (s, ϵ)-resilient sPRF if it is a ϵ-secure sPRF even if the attacker learns s bits of the n bit key.*

Now we can state the following lemma for the MD-then-Chop construction in terms of this assumption.

Lemma 10. *The MD-then-Chop construction* H_{chop_s} *is a* $\mathcal{O}(\epsilon'_{reg} + \ell \cdot \epsilon'_{sprf})$ *secure PRF (with PRF key prepended to the input) if the following conditions hold:*

- h *is a* (s, ϵ'_{sprf})*-resilient sPRF.*
- $h(U_n, \cdot)$ *is a* ϵ'_{reg} *output regular function.*

NMAC/HMAC construction.

The NMAC and HMAC constructions were shown to be secure arbitrary-length input PRFs by Bellare [3]. In [3], it is shown that the HMAC construction with $\alpha_1 = \alpha_2 = \perp$ (i.e. with the same IV for both invocations of the plain MD construction) is a secure arbitrary-length input PRF if the underlying compression function satisfies both the standard and dual PRF security definitions. This is done by simply prepending a different κ-bit key to each invocation of the plain MD construction [6].

Lemma 11. *The NMAC (resp. HMAC) construction* H_{nmac} *(resp.* H_{hmac}*) is a* $\mathcal{O}(q^2\ell \cdot \epsilon_{sprf} + \epsilon_{dprf})$ *PRF (with a different* κ*-bit key prepended to the input in each call to the MD construction) for any* IV_1 *and* IV_2 *(resp.* α_1 *and* α_2*) if the following conditions hold:*

[6]if the same key is prepended in both invocations, then the construction is secure under a slightly stronger assumption, called security against *related-key attacks* in [5, 3]. We ignore this setting here

- *h is a ϵ_{sprf}-secure sPRF.*
- *h is a ϵ_{dprf}-secure dPRF.*

4.2.3 Random Oracle

We already analyzed each of the four modes of operation for indifferentiability from random oracle in the chapter 3. However, we will mention these results briefly for completeness. Note that, in this case we need to make an "idealized" assumption on the compression function. In particular, we will assume that the compression function is a *fixed-length input random oracle* (FIL-RO).

Plain MD construction.

The plain MD construction does not give an indifferentiable construction a random oracle from a FIL-RO. This is essentially because the plain MD construction is vulnerable to the extension attack, as shown in chapter 3.

Encode-then-MD construction.

The suffix-free MD construction H_{suf} is also vulnerable to the extension attack, and cannot be indifferentiable from RO. However, if we apply a prefix-free encoding to the input then the resulting prefix-free MD construction H_{pre} is no longer vulnerable to this attack. Indeed, as shown in the previous chapter, this construction is indifferentiable from RO if the compression function is a FIL-RO.

MD-then-Chop construction.

The MD-then-Chop construction can be shown to be indifferentiable from RO if we chop a non-negligible (i.e. super-logarithmic in the security parameter) number of output bits. However, as shown in chapter 3, this construction has slightly worse exact security. In particular, we need a birthday bound over s (number of chopped) bits instead of n bits.

NMAC/HMAC construction.

The HMAC construction with $\alpha_1 = 0^\kappa$ and $\alpha_2 = \perp$ is indifferentiable from RO. We note that α_1 can be any κ-bit block such that $\alpha_1 \notin \{\perp, \alpha_2\}$, while α_2 can be any bit string in $\{\perp\} \cup \{0,1\}^\kappa$. On the other hand, the NMAC construction is indifferentiable from RO if $IV_1 \neq IV_2$.

4.2.4 Message Authentication Code

We will refer to MACs that work for fixed-length messages as FIL-MACs and those that work for variable-length messages as VIL-MACs. We will analyze each of the modes of operation to see if they satisfy VIL-MAC security. Let us first note, that a pseudorandom function can be considered to be a MAC as well (PRF output serves as the message tag). Thus all the results for PRF security above hold for VIL-MAC security as well. We will try to find if these modes are VIL-MACs under weaker assumptions on the compression function than those needed for the case of PRFs.

However, if we assume the compression function to be simply a FIL-MAC, then we cannot use the output of one application of the compression function to key the construction. One solution to this problem would be to analyze the construction in the *dedicated-key setting*, where each call to the compression function has a separate key space. For current hash functions, one could assume that part of the message block space can be used to securely key the compression function. That is, for the compression function $h : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$, the key occupies part of the first κ bits in the input. In this case, we can use the results of [1, 53] to get secure VIL-MAC constructions. However, as we discussed earlier, this violates the property that our modes of operation should be efficient in terms of input bandwidth. Thus, we will take a different approach here.

Plain MD construction.

If we prepend the MAC key $K \in \{0,1\}^\kappa$ to the input and apply the plain MD construction, then the resulting construction is vulnerable to the extension attack since the attacker can obtain the tag for a message by first getting a tag for the prefix. On the other hand, if the MAC key is appended to the input, then we find sufficient assumptions to show that H is a secure VIL-MAC. In particular, we will need the plain MD construction to be collision-resistant and the compression function to be a secure MAC when the MAC key occupies the first κ bits of its input.

Lemma 12. *The plain MD construction* H *is a* $\mathcal{O}(\ell\cdot(\epsilon_{reg}+\epsilon_{col})+\epsilon_{mac})$*-secure VIL-MAC, when the key is appended to the input, if the following conditions hold:*

- *h is a ϵ_{col} collision resistant function.*
- *h is a ϵ_{reg} output regular function.*
- *h is a ϵ_{mac} secure MAC, when the first κ bits of its input is considered to be the key space.*

Proof: The collision resistance of the plain MD construction (which we get from the first two conditions in the statement of the lemma) implies that the n-bit input to the last application of h is unique for each new input. Hence MAC security of the compression function h implies MAC security of the plain MD construction H. ◻

Encode-then-MD construction.

If we use a suffix-free encoding and append the MAC key to the input, then we succeed in reducing the assumptions needed in lemma 12 for collision resistance. Indeed, in this case, we can show that the suffix-free MD construction H_{suf} is $\mathcal{O}(\epsilon_{mac}+\ell\cdot\epsilon_{col})$ secure VIL-MAC if the compression function is ϵ_{col} collision resistant and ϵ_{mac} secure FIL-MAC. On the other hand, if we prepend the MAC key to the input, then the resulting construction is insecure since it is still vulnerable to the extension attack.

If we use a prefix-free encoding, and prepend the MAC key to the input then the resulting construction is a secure VIL-MAC only if all the conditions stated in lemma 9 hold. On the other hand, the prefix-free MD construction H_{pre} with MAC key appended to the input essentially has the same security as the plain MD construction in lemma 12.

MD-then-Chop construction.

If we prepend the MAC key to the input to the MD-then-Chop construction H_{chop_s}, then the resulting construction can be shown to be a VIL-MAC only under the conditions from lemma 10. On the other hand, if we append the MAC key to the input then we can prove the VIL-MAC security of the resulting construction by making slightly stronger assumptions on the compression function as compared to lemma 12.

Lemma 13. *The MD-then-Chop construction H_{chop_s} is mathcal$O(\ell \cdot (\epsilon_{reg} + \epsilon_{col}) + \epsilon'_{mac})$-secure VIL-MAC if the following conditions hold:*

- *h is a ϵ_{col} collision resistant function.*
- *h is a ϵ_{reg} output regular function.*
- *$h'(\cdot) = h(\cdot)|_{n-s}$ is a ϵ'_{mac} secure FIL-MAC.*

NMAC/HMAC construction.

In this case, if we prepend the MAC key to the input, then we need the same conditions as lemma 11 in order to prove VIL-MAC security as well. On the other hand, if we append the MAC key to the input, then both NMAC and HMAC constructions can only be proven secure using the same conditions as in the case of plain MD construction (lemma 12).

4.2.5 Target Collision Resistance

Target collision resistance (TCR) is a strictly weaker property than collision resistance. However, for some purposes, TCR hash functions (also called UOWHFs) suffice instead of CRHFs. For instance, it is possible to come up with a signature scheme on arbitrary length messages using one that works only for fixed-length messages by using TCR hash functions. For this reason, this primitive has attracted even greater interest since the discovery of better attacks against the collision resistance of existing hash functions.

When analyzing the TCR security of the hash function modes of operation, we cannot assume that the underlying compression function is a TCR function as well. This is because the output of a TCR function need not be random, so that each subsequent application of the compression function will require separate key space (and this dedicated-key setting violates our requirements from the mode of operation). Instead, we will assume that the compression function $h : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$ is an unkeyed function that satisfies second preimage resistance type properties.

Plain MD construction.

In order to discuss the TCR security of the plain MD construction, we need to first discuss appropriate keying mechanisms for this construction. As we briefly mentioned above, Shoup [70] described an efficient *masking-based construction* based on the plain MD construction. However, this construction modifies the chaining variable which violates our properties of black-box modes of operation. Unfortunately, we do not know of any black-box ways of keying the plain MD construction such that it can be shown to be a TCR hash function only assuming the compression function to be a SPR function.

Halevi and Krawczyk [40] suggested an alternate way of keying the plain MD construction that satisfies all the properties of a black-box mode of operation. The construction H_K proposed in [40] uses a κ-bit key K and XORs the key with each message block in the plain MD construction, i.e.

$$\mathsf{H}_K(x_1 \parallel \ldots \parallel x_\ell) \overset{def}{=} h(K \oplus x_\ell, h(\ldots, h(K \oplus x_1, IV) \ldots))$$

However, in order to prove TCR security of this construction one needs to make a slightly non-standard "SPR-type" assumption on the compression function, called the *evaluated SPR assumption* (e-SPR) [40].

Assumption 3 (evaluated second preimage-resistance). *Consider a function $h : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ and let H_K be the plain MD based* *construction using h (described above). The function h is ϵ* evaluated second preimage resistant *if any efficient machine A wins in the following game with probability at most ϵ (over the random choice of IV and the coins of A).*

1. *A chooses a sequence of κ-bit blocks $\Delta_1, \ldots, \Delta_i$.*

2. *The challenger chooses a random key K and sets $c = \mathsf{H}_K(\Delta_1 \oplus K, \ldots, \Delta_{i-1} \oplus K)$ and $m = \Delta_i \oplus K$.*

3. *A wins if it can find c' and m' such that $h(m', c') = h(m, c)$.*

Halevi and Krawczyk [40] show that if the compression function h is an e-SPR function, then the construction H_K described above is a secure TCR hash function. However, in their proof they require that the inputs provided to H_K must be suffix-free. Indeed, this is required for their reduction to go through. However, we note that even for the plain MD construction (with possibly "non-suffix-free" inputs), one can make an additional assumption on the compression function to enable us to apply the proof technique of [40].

Lemma 14. *The construction H_K is an $\mathcal{O}(\ell \cdot (\epsilon_{fix} + \epsilon_{espr}))$-secure TCR hash function if the following conditions hold:*

- *h is an ϵ_{espr}-secure e-SPR function.*

- *For a randomly chosen IV, no efficient machine A has success probability more than ϵ_{fix} in finding a sequence of κ-bit blocks such that:*

$$h(x_i, h(\ldots, h(IV, x_1)\ldots)) = IV$$

Encode-then-MD construction.

If we apply a suffix-free encoding to the input before using the construction, then the resulting mode of operation $\mathsf{H}_{suf,K}$ is a TCR hash function based only on the assumption that h is an e-SPR function [40]. On the other hand, if we use a prefix-free encoding then it does not help in improving the security of the plain MD construction and we need all conditions of lemma 14 to prove the TCR security of the resulting construction.

MD-then-Chop construction.

For the MD-then-Chop construction, we need to make a slightly stronger assumption on the compression function to prove the TCR security of the resulting construction. In particular, we need to assume that the compression function h is e-SPR even if we chop a non-negligible number of its output bits. If we replace the second condition in lemma 14 wit this stronger condition, then it holds for the MD-then-Chop construction as well.

NMAC/HMAC construction.

Using the NMAC or HMAC construction does not lead to improved TCR security of the resulting construction. Again this is because if the attacker finds a collision in the first invocation of the plain MD construction then it implies a collision for both NMAC and HMAC construction.

4.2.6 Second Preimage Resistance.

In this section, we will analyze each of the modes of operation for the minimal assumptions on the compression function needed in order to prove the SPR security of the construction. Unfortunately, to the best of our knowledge, there is no black-box mode of operation that is property preserving for second preimage resistance. Hence we will need to make a slightly stronger assumption on the compression function h.

Assumption 4 (computed Second Preimage Resistance (cSPR) [40]). *A function $h : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$ is a ϵ-*secure cSPR function *if any efficient machine A has success probability at most ϵ in the following game:*

1. *The challenger randomly selects a sequence of κ-bit blocks $x_1, \ldots, x_\ell$, sets $c = \mathsf{H}(x_1 \parallel \ldots \parallel x_{i-1})$ and $x = x_i$. Here H is the plain MD construction using the compression function h and random IV. The challenger sends $x_1, \ldots, x_i$ to A.*

2. *A wins if it finds $(x', c') \in \{0,1\}^\kappa \times \{0,1\}^n$ such that $h(x', c') = h(x, c)$.*

Note that this assumption is quite similar to the eSPR assumption that we needed for the TCR hash function case.

Plain MD construction.

Here we will assume that the compression function h is a cSPR function. However, in order to prove SPR security of the plain MD construction, we will also need to assume that the attacker cannot find fixed points starting from a random IV.

Lemma 15. *The plain MD construction* H *using the compression function* h *is a* $\mathcal{O}(\ell \cdot (\epsilon_{cspr} + \epsilon_{fix}))$*-secure SPR function if the following conditions hold:*

- h *is a* ϵ_{cspr}*-secure cSPR function.*
- *For a random initialization vector* IV*, no efficient attacker can find a sequence of* κ*-bit blocks* $x_1, \ldots, x_i$ *such that* $\mathsf{H}(x_1 \parallel \ldots \parallel x_i) = IV$ *with probability more than* ϵ_{fix}.

Encode-then-MD construction.

If we use a suffix-free encoding on the input, then the resulting construction H_{suf} can be proven to be a SPR function solely on the assumption that the compression function h is a cSPR function. On the other hand, the prefix-free MD construction does not help in gaining any improvement in SPR security over the plain MD construction.

Lemma 16 ([40]). *The suffix-free MD construction* H_{suf} *is a* $\mathcal{O}(\ell \cdot \epsilon_{cspr})$*-secure SPR function if the compression function* h *is a* ϵ_{cspr}*-secure cSPR function.*

MD-then-Chop construction.

As in the case of collision resistance and TCR functions, we will need to impose a slightly stronger assumption on the compression function in order to prove the SPR security of the MD-then-Chop construction H_{chop_s}. In particular, we will need to assume that the function $h'(\cdot) = h(\cdot)|_{n-s}$ is ϵ'_{cspr}-secure SPR function. This assumption along with the second condition from lemma 15 suffices to show that the MD-then-Chop construction is a $\mathcal{O}(\ell \cdot (\epsilon_{fix} + \epsilon'_{cspr}))$-secure SPR function.

NMAC/HMAC construction.

The NMAC/HMAC construction do not give any better SPR security as compared to the plain MD construction. This is because a collision in the first invocation of the MD construction implies a collision for both the NMAC and HMAC constructions.

4.2.7 Randomness Extraction

The idea of using the MD construction as a randomness extractor was discussed by Dodis et al in [25]. They showed that for getting any useful randomness extraction properties from the MD construction, one needs to make a really strong assumption on the compression function h. In particular, they assume that the compression function $h : \{0,1\}^{\kappa} \times \{0,1\}^n \to \{0,1\}^n$ is an *ideal randomness extractor*, which is the same as assuming it to be a *family of random functions*[7]. That is, the function $h(\cdot, x)$ is a random function from κ to n bits when x is uniformly distributed. We debate such a compression

[7]Note that this is a weaker assumption than assuming h to be a FIL-RO. In particular, it is a (very inefficiently) realizable assumption.

function h as a family of random functions $\{h_r\}$ for $r \in \{0,1\}^n$.

Let us start by explaining why one needs to make such a strong assumption on h. If we assume h to be a regular extractor, then the distribution of the output of $h(\cdot, x)$ for random x has a non-zero statistical distance from the uniform distribution on $\{0,1\}^n$. If this output is used a seed for the next application of the compression function then one has no guarantee of extraction, since the seed us no longer independent of the κ-bit input block. Actually, Dodis et al [25] do give a positive result for the MD construction simply under the assumption that h is an *almost-universal family of functions* [8]. However, for this result they require that every input block for the MD construction must have some amount of *conditional min-entropy*[9] (see [25] for more details). However, all the results here are based on the assumption that h is an ideal randomness extractor.

Plain MD construction.

In this case, one can show that for a restricted class of inputs (from certain high min entropy distributions), the output of the plain MD construction, using an *ideal randomness extractor* h, is close to uniform. The input distribution should be such that it has high overall min entropy as well as high conditional min-entropy in the last input block.

[8] i.e., for the function $h(\cdot, x)$, where x uniformly distributed on $\{0,1\}^n$, for any two distinct inputs y and z, the probability that $h(z,x) = h(y,x)$ is negligibly close to the corresponding probability for a random function

[9] This is the min entropy of an input block conditioned on all the other input blocks.

Lemma 17 ([25]). *Let $\{\mathsf{H}_r\}$ be the plain MD construction defined over a family of random functions $\{h_r\}$, where the seed r is essentially the random IV in the plain MD construction. Let $\mathcal{X}$ be the distribution of the inputs to* H *(over bit strings with at most ℓ κ-bit blocks) and let $\mathcal{X}_\ell$ be the distribution induced by $\mathcal{X}$ on the last block of the input. If $H_\infty(\mathcal{X}) > n + 2\log\left(\frac{1}{\epsilon}\right)$ and $H_\infty(\mathcal{X}_\ell) > \log \ell + 2\log\left(\frac{1}{\epsilon}\right)$, then $\mathbf{SD}(\mathsf{H}_{\mathbf{IV}}(\mathcal{X}), \mathbf{U_n}) = \mathcal{O}(\epsilon)$ where U_n is the uniform distribution on n-bit strings.*

Encode-then-MD construction.

Note that if one applies a suffix-free encoding to the input in conjunction with the plain MD construction, then the (encoded) input to the MD construction may no longer satisfy the min entropy requirements from lemma 17. Indeed, consider applying *Merkle-Damgård strengthening* to the input before the MD construction. In this case, the last block has no conditional entropy (since it is simply the input length). Nonetheless, in [25], Dodis et al. show that adding any fixed padding to an input that satisfies all min entropy requirements still gives a good randomness extractor! Similarly, we cannot say much about a general prefix-free encoding since it might change the input distribution in an arbitrary way. However, if we consider prepending input length to the input, then it still gives a good randomness extractor.

MD-then-Chop construction.

Quite surprisingly, if we chop a sufficient number of output bits then one can prove randomness extraction properties of the resulting construction based on fewer assumption than lemma 17. In particular, we can get rid of the requirement that the last input block has sufficient conditional min entropy.

Lemma 18 ([25]). *Let $\mathsf{H}_{chop_s,r}$ be the MD-then-Chop construction defined over a family of random functions $\{h_r\}$, where the seed r is essentially the random IV used in the construction. Let $\mathcal{X}$ be the input distribution to H_{chop_s} (over bit strings with at most ℓ k-bit blocks). If $H_\infty(\mathcal{X}) = n + s + \log(\ell + 1)$, then we get that $\mathbf{SD}(\mathsf{H}_{chop_s}(\mathcal{X}), U_{n-s}) \leq 2^{-s}$ where U_{n-s} is the uniform distribution on $(n-s)$-bit strings.*

NMAC/HMAC construction.

For the NMAC construction, it can be shown that if random and independent IV_1 and IV_2 are used in the two applications of the plain MD construction, then the resulting construction H_{nmac} is a good randomness extractor if the compression function represents a family of random functions. We can then restate lemma 17 for the construction H_{nmac} as well, with the same exact security. However, it turns out that translating these results to the setting of the HMAC construction is not straightforward [25].

4.2.8 One-Wayness

One way functions are also often referred to as preimage resistant functions. This security property is even weaker than second preimage resistance.

Plain MD construction.

In this case, we will need to assume that the compression function h is a one way function. Moreover, we will also require that h is output regular, so that its output is uniformly distributed for a random input. This is essentially because we need the input to a one-way function to be random in order to use the one-wayness property.

Lemma 19. *The plain MD construction* H *is* $\mathcal{O}(\ell \cdot \epsilon_{reg} + \epsilon_{owf})$*-secure one-way function if the following conditions hold:*

- *h is an ϵ_{reg} output regular function.*
- *h is a ϵ_{owf}-secure one-way function.*

Proof: Say an attacker A has non-negligible advantage in the one-wayness game against the plain MD construction H. Then we can construct another attacker A', that simply gives its challenge as a challenge to A. Since h is output regular, the attacker A cannot tell the difference between this challenge and if it was given the output of H on a random input. Thus it has non-negligible success probability in this game as well. The attacker A' can use the preimage outputted by A to invert its challenge as well. $\square$

Encode-then-MD construction.

Note that if we use an arbitrary suffix-free encoding before the MD construction, then we cannot say much about the one-wayness of the resulting construction since the input distribution could be arbitrary. However, if we apply *Merkle-Damgård strengthening* to the input, then we can show that the resulting construction is a one-way function under sufficient assumptions. In particular, we need to make an additional assumption that for any message block, most outputs of the compression function have a small number of preimages in the chaining variable that are consistent with the given block. Note that this property certainly holds for a random compression function (and, thus, holds for most compression functions). As for prefix-free encoding, once again we cannot say anything general (for the same reason as above), but when prepending the message length we are essentially back to the setting of plain MD discussed above, except we need to assume that the output of the compression function on a random IV and a fixed message block is random.

In particular, we note that encoding the input in any way does not help as far as one-wayness of the construction is concerned. In fact, we only need more assumptions to prove this property, as compared to the plain MD construction.

MD-then-Chop construction.

In order to prove the one-way security of the MD-then-Chop construction, we will need to make a slightly stronger assumption on the compression function h. In particular, we assume that the compression function h is one-way with s bits of the output chopped. Let the one-way security of the function h with truncated output be ϵ'_{owf}. Then we can show that H_{chop_s} is a $\mathcal{O}(\ell \cdot \epsilon_{reg} + \epsilon'_{owf})$-secure one-way function (similar to lemma 19)

NMAC/HMAC construction.

The NMAC construction is a one-way function under the same conditions on the underlying compression function h as required in lemma 19. However, we require that random and independent initialization vectors IV_1 and IV_2 are used in the NMAC construction. However, it turns out that translating these results to the setting of the HMAC construction is not straightforward.

4.3 Implications for Actual Hash Functions

We will now translate our results into suggestions for usage of actual "cascade construction based" hash functions, such as functions from the SHA family. As we mentioned earlier, we have tried to find the minimal assumptions needed to make each of the four modes of operation secure (for each of the security properties). Thus, we have left part of the "decision making" for the practitioner who uses our results. In particular, the practitioner must consider the following questions:

1. What one needs to assume about the hash function in order for the cryptosystem (that the hash function is being used for) to be provably secure?

2. What level of trust the practitioner is willing to place in the underlying compression function?

The answer to the first question will help in deciding the security property to look for in the hash function mode of operation. The answer to the second question may not be as straightforward since the design of the compression functions is quite complex and mostly based on heuristic. In this case, the practitioner needs to weigh all the properties (s)he desires from the cryptosystem, in terms of efficiency, security etc. Thus, while some may be willing to make a slightly stronger assumption on the compression function to have a more efficient implementation, others may be willing to sacrifice some efficiency for better security.

Now we will give some basic recommendations for actual hash functions with respect to the various security properties.

4.3.1 Collision Resistance

Each of the SHA functions are essentially based on the suffix-free MD construction (using MD strengthening). Hence, collision resistance for each of these hash functions is asymptotically same as finding collisions on the compression function. It does not make much sense to use the "truncated" versions, SHA-224 and SHA-384, since this only sacrifices the collision resistance of the original "untruncated" version (i.e. SHA-256 and SHA-512, respectively). Using the NMAC/HMAC construction does not help in this case.

4.3.2 Pseudorandomness

We note that using the full SHA-256 or SHA-512 hash functions makes more sense for pseudorandomness than using the chopped versions (SHA-228 or SHA-384), which only have worse security. If any of the SHA functions are used, as it is, for pseudorandomness, then we recommend appending the PRF key to the input instead of prepending it. However, we recommend using these functions in conjunction with a prefix-free encoding (such as prepending input length to the input) in which case the PRF key should be *prepended* to the input. Another option would be to compose two calls to SHA-1, with independent keys prepended in each call, to get security based on the sPRF and dPRF security of the compression function.

4.3.3 Random Oracle

Note that none of the SHA functions should be used, as it is, if the security of the cryptosystem requires the *random oracle assumption* for the hash function. This is because the plain MD construction (even with MD strengthening) is vulnerable to simple attacks in the indifferentiability scenario. One may think that both SHA-224 and SHA-384 that correspond to "chop" versions of the functions SHA-256 and SHA-512 would be secure (since the MD-then-Chop construction is secure). However, note that only 32 bits are chopped in the case of SHA-224, which does not give sufficient security for almost all applications. Hence, only SHA-384 (that chops 128 bits) may be suitable to be used directly to instantiate the random oracle.

We recommend using the HMAC construction involving two black-box calls to the SHA function (while prepending different α_1 and α_2 in each cal) for this purpose. Using any of these hash functions in conjunction with a prefix-free encoding will also work for this purpose.

4.3.4 Message Authentication

If the SHA functions are used as MACs directly, then the MAC key should be appended to the input. In this case, security depends on both the MAC security and collision resistance of the compression function. Using the HMAC construction does not help in improving the security either. Moreover, when the "chopped" functions SHA-224 or SHA-384 are used as MACs, then their security is only worse than the unchopped versions (SHA-256 and SHA-512).

If one is willing to assume pseudorandomness of the compression function, then the techniques mentioned above for pseudorandomness can be used as well. Another approach would be to assume the *dedicated-key setting*, by inserting the MAC key in each application of the compression function (at the cost of some input bandwidth) and then one could use one of the techniques suggested in [1, 53].

4.3.5 Target Collision Resistance or UOWHFs

We recommend using the technique suggested by Halevi and Krawczyk [40] if the SHA functions are used as UOWHFs. In this case, one XORs the UOWHF key to each block of the input. Since MD strengthening is already used in all these functions, the UOWHF security of this construction is based only on the eSPR [40] (see above) of the compression function.

4.3.6 Second Preimage Resistance

It makes sense to use the SHA hash functions directly for the purpose of *second preimage resistance* without using any additional techniques, since they do not lead to improved security (note that these functions already incorporate MD strengthening).

4.3.7 Randomness Extraction

All the positive results for *randomness extraction* have reasonable interpretation in practice, only if we are willing to assume that the SHA compression function is close to being a family of random functions. Even though it is theoretically impossible, since the SHA compression function has a short description, it might still be a more reasonable assumption than assuming the compression function to be a FIL-RO.

Under this assumption, we can deduce that the SHA functions are good randomness extractors for input distributions with high min entropy overall and in the last block. On the other hand, as we saw above, it might be a good idea to use chopped function SHA-384 for this purpose to get better extraction properties (SHA-224 does not have sufficient number of chopped bits to give useful advantage). Using the HMAC construction does not help in improving the extraction properties.

4.3.8 One-Wayness

In the case of “one-wayness”, the security of the chopped functions, SHA-224 and SHA-384, seems to rely on stronger assumptions than the security of the corresponding “unchopped” versions (SHA-256 and SHA-384). This is because the one-way security increases with the number of output bits. On the other hand, it might be the case that SHA-224 still has higher security than SHA-1, which seems intuitive given the bigger IV of SHA-224. Moreover, message encoding or HMAC construction only seems to decrease the one-wayness of the hash function.

Part II

Block Ciphers

Chapter 5

Feistel network made public

Feistel Networks are extremely popular tools in designing "cryptographically strong" functions. Such networks are based on iterative application of the simple *Feistel permutation*. In particular, given a function $f : \{0,1\}^n \to \{0,1\}^n$, the Feistel permutation based on function f is defined as:

$$\begin{aligned} \Psi_f : \{0,1\}^{2n} &\to \{0,1\}^2 n \\ x_L \parallel x_R &\mapsto x_R \parallel f(x_R) \oplus x_L \end{aligned}$$

Typically, a *Feistel network* consists of several iterative applications of the Feistel permutation with independent functions f used in each application. The various iterative applications of the Feistel permutation are called the *rounds* of the Feistel network, while the corresponding functions are called *round functions*. Among their applications, they are commonly used in the design of popular block ciphers, such as DES, as well as other constructs, such as popular padding schemes OAEP [9] or PSS-R [10]. Even though a theoretical justification for the use of the Feistel network in design of DES was not disclosed at the time of its release, this justification was later provided by the celebrated result of Luby and Rackoff [47].

5.0.9 Luby-Rackoff's result and Improvements

Luby and Rackoff's result [47]. Luby and Rackoff noted that the security of a block cipher can be best analyzed in terms of its "closeness" from a uniformly random permutation for each key. That is, it should be an independent *pseudorandom permutation* for every different key. Moreover, as a justification for the use of Feistel network in the design of block ciphers, they showed that three (resp. four) rounds of the Feistel transform are sufficient to turn a pseudorandom function (PRF) family into a pseudorandom permutation (PRP) family (resp. strong PRP family (SPRP)). In particular, their construction of a PRP (resp. SPRP) consisted of a three (resp. four) round Feistel network with independent PRFs (from the PRF family) as round functions.

Subsequent Improvements. There has been a lot of subsequent work on improving various aspects of the Luby-Rackoff result. (referred to as "LR" from now on). Naor and Reingold [58] provided a simpler proof of the LR result. Moreover, they generalized the result and showed that the four round construction remains secure even if the first and last round of the Feistel transform are replaced by pairwise independent permutations.

Maurer and Pietrzak [51] studied the exact security of the LR construction if the number of Feistel rounds is increased from four. In particular, they noted that LR type proofs consist of two parts, first the PRFs in each rounds of the PRP/SPRP construction are replaced by uniformly random functions. Then the resulting construction is proven to be secure against any unbounded attacker that makes less than an exponential (of the form $\mathcal{O}(2^{cn})$) number of queries. LR showed that the four round construction is secure against any attacker that makes $\mathcal{O}(2^{n/2})$ queries. Maurer and Pietrzak [51] improved this result by showing that a $6k$ round Feistel network is secure against any attacker that makes upto $\mathcal{O}(n \cdot (1 - O(1/k)))$ queries (thus approaching the information-theoretic bound of $\mathcal{O}(2^n)$ as $k \to \infty$).

Patarin [64] significantly improved this result by showing that a 5 (resp. 6) round Feistel network is a secure PRP (resp. SPRP) against any unbounded attacker making $q \ll 2^n$ queries, thus showing that the information-theoretic bound can be achieved within a constant number of rounds.

Ramzan and Reyzin [69] generalized the LR result from a different perspective. They studied the security of the Feistel network (with PRFs) against attackers that are given oracle access to some (or all) of the round functions. They showed that if the attacker has oracle access to the middle two round functions, then the four round Feistel network is still a secure SPRP. On the other hand, they also showed that if there is an efficient attacker that, when given oracle access to either of the first or last round function, breaks the SPRP security of the four round construction.

Apart from these there were results that studied the security of the Feistel network when not all of the round functions are independent ([67, 63]). In a recent work, Maurer et al [50] studied the security of the Feistel network when the round functions are non-adaptively secure PRFs instead of adaptive security.

5.0.10 The Problem and Our Result

A common aspect of all the works on the Feistel network mentioned above is that they crucially rely on the following assumptions:

(a) *the (pseudo)randomness of round functions*; and

(b) *the secrecy of (at least some of) the intermediate round values appearing during the Feistel computation*

If either of these assumptions is not true then each of the above results is no longer valid. However, there are several natural scenarios where one (or both) of these assumptions are violated.

Is Unpredictability enough? We start with the assumption regarding pseudorandomness of round functions. The assumption is quite strong, since practical block ciphers certainly do not use PRFs as their round functions. Instead, they heuristically use considerably more than the three-six rounds predicted by the LR and all subsequent "theoretical justifications". Thus, a large disconnect still remains to be bridged. Clearly, though, we need to assume some security property of the round functions, but can a weaker property be enough to guarantee security? In the context of domain extension of message authentication codes, An and Bellare [1] studied a natural question whether *unpredictability* — a much weaker property than pseudorandomness — can at least guarantee the unpredictability of the resulting Feistel permutation. Although not as strong as pseudorandomness, this will at least guarantee some minimal security of block ciphers (see next chapter), is enough for message authentication, and anyways doubles the domain of the unpredictable function, which is useful (and non-trivial!) by itself. [1] gave a negative answer for the case of three rounds, and suggested that "even more rounds do not appear to help". This result indicates that previous "LR-type techniques" are insufficient to handle unpredictability (since in the case of PRFs three rounds are enough), and also leaves open the question whether more Feistel rounds will eventually be enough to preserve unpredictability.

Is it Safe to Leak Intermediate Results? Another crucial reason for the validity of the LR result is the fact that all the intermediate round values are never leaked to the attacker. In fact, the *key* to the argument, and other results mentioned above, is that the attacker effectively gets no information about most of these values in case a PRF is used for the round functions, and simple attacks (which we later generalize to many more rounds) are possible to invalidate the LR result in case the intermediate round values are leaked. Unfortunately, for many natural applications this assumption (or conclusion!) can not be enforced, and totally new argument is needed. There are several examples of such applications.

Starting with the simplest example, intermediate values may be inadvertently leaked through an attack. For example, one might imagine a smartcard implementing a block cipher via the Feistel network using a secure chip implementing a PRF. In this case, the attacker might be able to observe the communication between the smartcard and the chip, although it is unable to break security of the chop. More realistically, when round functions are not PRFs, the attacker might get a lot of information about the intermediate values anyway, even without extra attack capabilities. For instance, in the case of unpredictable functions (UFs) mentioned above, we will construct provably secure UFs such that the output of the Feistel Network completely leaks all intermediate round values. Although artificial, this example illustrates that weaker assumptions on the round functions can no longer guarantee the secrecy of intermediate values. For yet another example, the round function might simply be public to begin with. This happens when one considers the question of implementing an ideal cipher from a random oracle. In this case the round function is a publicly accessible random oracle, and is certainly freely available to the attacker. This question will be considered in chapter 7. As a final example (not considered in prior work), the attacker might get hold of the intermediate values because the *application requires to reveal such values.* This happens when one tries to add *verifiability* to PRFs and

PRPs (or their unpredictable analogs), which we describe in more detail in the next chapter.

Our Results. In order to deal with such situations when the intermediate round values may be leaked to the adversary, and also handle cases when the round functions may not be pseudorandom, we develop a new understanding of the Feistel network. In particular, we develop a general framework for studying the Feistel network that is applicable in all such scenarios. In our modeling, a k-round Feistel network is applied to k members $f_1 \dots f_k$ independently selected from some (not necessarily pseudorandom) function family C, resulting in a Feistel permutation π. Whenever an attacker makes a forward (resp. inverse) query to π (resp. π^{-1}), we assume that it learns all intermediate values.

On the negative side, we show a simple attack allowing an adversary to compute any value $\pi^{-1}(y)$ by making at most exponential in k number of *forward* queries to π. Since such an inversion should be unlikely (with polynomially many queries) even for an unpredictable permutation, this immediately means that at least a superlogarithmic number of Feistel rounds (in the security parameter λ) are *necessary* to guarantee security for *any* of the applications above. Aside from showing the *tightness of all our positive result* describe below, this result partially explains *why practical block ciphers use significantly more than* $3-6$ *rounds* predicted by all the previous "theoretical justifications" of the Feistel network. Indeed, since all such ciphers heuristically use round functions which are not PRFs, and we just showed that even unpredictable functions might leak a lot (or even *all*) of the intermediate results, the simple attack we present might have been quite applicable if a small constant number of rounds was used!

On the positive side, we show a general combinatorial property of the Feistel Network which makes essentially no assumptions (such as pseudorandomness) about the round functions used in the Feistel construction, and allows us to apply it to a wide variety of situations described above, where the previous techniques failed. In essence, for any $s \leq k/2$, we show that if an attacker, making a sub-exponential in s number of (forward or backward) queries to the construction and always learning all the intermediate round values, can cause a non-trivial collision somewhere between rounds s and $k-s$, then the attacker can also find a simple (and non-trivial) XOR condition on a constant (up to six) number of the round values of the queries he has made. This means that if a function family C is such that it is provably hard for an efficient attacker to find such a non-trivial XOR condition, — and we call such families *5-XOR resistant*, — then it is very unlikely that the attacker can cause any collisions between rounds s and $k-s$ (as long as s, and thus k, are super-logarithmic in the security parameter λ). In the next chapter, we show that once no such collisions are possible, it is possible to directly argue the security of the Feistel Network for our applications.

5.1 Preliminaries

We will denote by $\mathsf{Fibonacci}(k)$ the k^{th} Fibonacci number, and thus $\mathsf{Fibonacci}(k) = \mathcal{O}(1.618^k)$.

The *Feistel transformation* using $f : \{0,1\}^n \to \{0,1\}^n$ is a permutation Ψ_f on $2n$ bits defined as, $\Psi_f(x) \overset{def}{=} x_R \parallel x_L \oplus f(x_R)$. The symbols x_L and x_R denote the left and right halves of the $2n$ bit string x. We will also call the Feistel network consisting of k iterated applications of the Feistel transformation, a k-round LR construction and denote it by $\Psi_{f_1 \cdots f_k}$ (or Ψ_k when $f_1 \dots f_k$ are clear from context) where $f_1 \dots f_k$ are the round functions used. On a $2n$ bit input, the construction Ψ_k generates $(k+2)$ n-bit intermediate round values, the last two of which form the output. This construction is illustrated in figure 5.1.

5-XOR CONDITION. Consider a k-round LR construction Ψ_k, that uses arbitrary round functions $f_1 \dots f_k$. Now consider any sequence of q forward/inverse queries provided to this construction. As discussed above, in the process of computing its output the LR construction Ψ_k also generates $(k+2)$ n-bit round values. We will denote the n-bit round values associated with the i^{th} query as $R_0^i, R_1^i, \dots, R_k^i, R_{k+1}^i$. If this was forward (resp. inverse) query, then the $2n$ bit input value is $R_0^i \parallel R_1^i$ (resp. $R_k^i \parallel R_{k+1}^i$).

For any $j \in \{1, \dots, k\}$, the LR construction performs the round function evaluation $f_j(R_j^i)$. We call this a *new round function evaluation* if $R_j^i \neq R_j^{i'}$ for any $i' < i$. In this case, if the i^{th} query is a forward (resp. inverse) one,

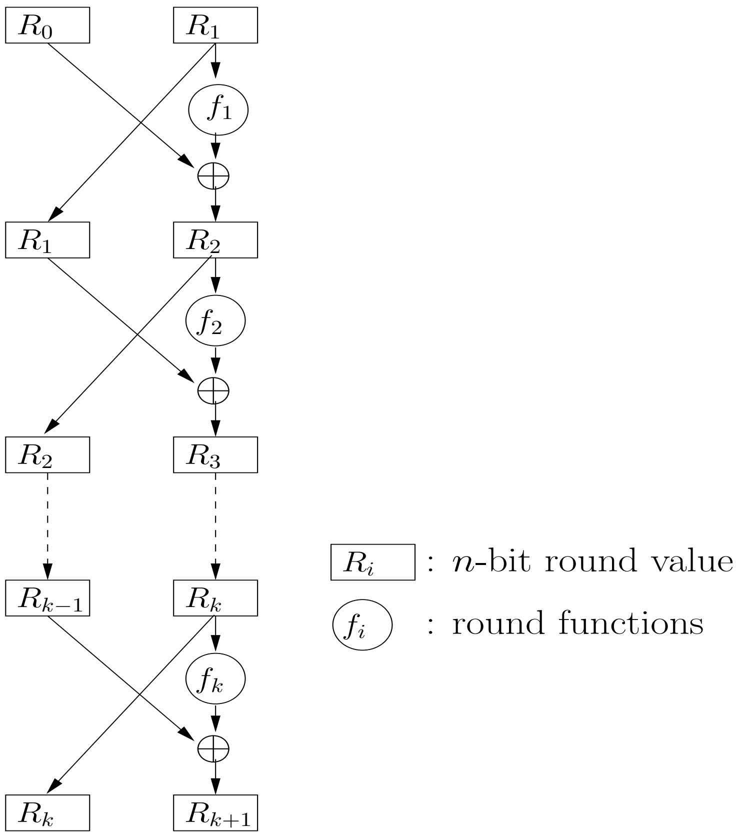


Figure 5.1: The k-round Feistel Network

then the round value R^i_{j+1} (resp. R^i_{j-1}) is the new round value generated as a result of this round function evaluation.

We say that the **5-XOR condition** holds for this sequence of q queries, with corresponding round values $\left\{R^i_0, R^i_1, \ldots R^i_{k+1}\right\}_{i\in\{1\ldots q\}}$, if there is at least one *new round function evaluation* such that the corresponding new round value R^i_j generated as a result can be represented as a bit-by-bit XOR of upto 5 previously existing round values, that is:

- If i^{th} query is a forward query, then R^i_j can be represented as an XOR of upto 5 round values $R^{i'}_{j'}$, such that either $i' < i$ or $(i' = i) \wedge (j' < j)$.
- If i^{th} query is an inverse query, then R^i_j can be represented as an XOR of upto 5 round values $R^{i'}_{j'}$, such that either $i' < i$ or $(i' = i) \wedge (j' > j)$.

5.2 Insecurity of $\mathcal{O}(\log \lambda)$-round Feistel

We will demonstrate here that upto a logarithmic number of Feistel rounds do not suffice for any of our results. Essentially, we will describe an efficient attacker A that only makes forward queries to the k-round LR construction Ψ_k (using arbitrary round functions $f_1 \ldots f_k$), and finds the input corresponding to any permutation output $y \in \{0,1\}^{2n}$.

Theorem 18. *For the k round LR construction Ψ_k that uses $k = \mathcal{O}(\log \lambda)$ round functions, there exists a probabilistic polynomial time adversary A_π that takes oracle access to Ψ_k. The adversary A_π makes $\mathcal{O}(\mathsf{Fibonacci}(k)) = \mathsf{poly}(\lambda)$ forward queries to Ψ_k and with high probability finds the input corresponding to an output y without actually making that query.*

Proof: The adversary A_π gets the permutation output $y \in \{0,1\}^{2n}$, that it is supposed to invert Ψ_k on. For concreteness, we assume that $y = 0^{2n}$ (anything else works just as well). We will describe the recursive subroutine that the attacker A_π is based on. Say the round functions of Ψ_k are $f_1 \ldots f_k$. The recursive function that we describe is $E(j, Y)$, where j is the number of rounds in the Feistel construction and Y is a $2n$ bit value, and the task of $E(j, Y)$ is to find the input such that the j^{th} and $(j+1)^{th}$ round values are Y_L and Y_R (the left and right halves of Y), respectively.

- $\mathbf{E(1,Y)}$: Choose a random $R'_0 \leftarrow \{0,1\}^n$. Make the forward query $R'_0 \parallel Y_L$ to Ψ_1, where the 2^{nd} round value is R'_2. Now the 1^{st} and 2^{nd} round values for the input $R'_2 \oplus R'_0 \oplus Y_R \parallel Y_L$ are Y_L and Y_R.

- $\mathbf{E(j,Y)}$, $\mathbf{j > 1}$: Perform the following steps,
  - Make a random query $R_0 \parallel R_1 \leftarrow \{0,1\}^{2n}$, and say the $2n$ bit value at the j^{th} round is is $R_j \parallel R_{j+1}$. Then, $f_j(R_j) = (R_{j-1} \oplus R_{j+1})$.
  - Run $E(j-2, (f_{j-1}(R_{j-1}) \oplus Y_L) \parallel R_{j-1})$ and the $2n$ bit value at the $(j-1)^{th}$ round is $R_{j-1} \parallel Y_L$. Hence $f_j(Y_L) = R_{j-1} \oplus R_{j+1}$.
  - Run $E((j-1), (f_j(Y_L) \oplus Y_R) \parallel Y_L)$, and the j^{th} and $(j+1)^{th}$ round values are Y_L and Y_R, respectively.

The adversary A_π essentially runs the algorithm $E(k, 0^{2n})$. Now we need to make sure that the adversary A_π does not query on the input corresponding to the output 0^{2n}. But since all the queries made in the recursive algorithm are essentially chosen at random, we know that the probability of this happening is $\frac{q}{2^{2n}}$. Hence, the probability that A_π succeeds is at least $\left(1 - \frac{q}{2^{2n}}\right)$. $\square$

We note that the above attacker works in a scenario where it can only make forward queries to the Feistel construction Ψ_k. In case, it can make inverse queries as well, it is possible to design a similar attacker that succeeds in

$\mathcal{O}(\mathsf{Fibonacci}(k/2))$ queries. If the number of rounds $k = \mathcal{O}(\log \lambda)$, then the number of queries needed by either of these attackers is polynomial in the security parameter λ. We will describe how this attacker works in each of the applications of the Feistel network in the next chapter.

5.3 Combinatorial Analysis of the Feistel Network

In this section, we will prove a general combinatorial lemma about the k-round LR construction Ψ_k, that uses arbitrary round functions $f_1 \dots f_k$. In the subsequent chapter, we will see that this lemma is the main ingredient in deriving each of our results.

Consider an arbitrary sequence of q forward/inverse permutation queries made to the LR construction Ψ_k, each of which is a $2n$ bit string. Denote the $(k+2)$ n-bit round values associated with the i^{th} query as $R[i,0], R[i,1], \dots,$ $R[i,k], R[i,(k+1)]$, where $R[i,0] \parallel R[i,1]$ (resp. $R[i,k], R[i,(k+1)]$) is the input if this is a forward (resp. inverse) query. We say that such a sequence of queries produces a s^{th} round value collision, if the s^{th} round value collides for two different permutation queries from this query sequence. That is, when we have $R[i,s] = R[j,s]$ for $i,j \in \{1 \dots q\}$ and $(R[i,0] \parallel R[i,1]) \neq (R[j,0] \parallel R[j,1])$.

We essentially show that if any such sequence of q queries produces a r^{th} round value collision for any $r \in \{s \dots (k-s)\}$ (where $s \leq (k/2)$), then one of the following must hold:

1. The number of queries q is exponential in s.
2. For this sequence of queries, the *5-XOR condition* holds.

5.3.1 A "No-Collision" Property of the Feistel Network

Now we will state the "no round value collision" property described above. This is formalized in the lemma below:

Lemma 20. *Let Ψ_k be a k round LR construction that uses fixed and arbitrary round functions $f_1 \ldots f_k$. For any $s \leq \frac{k}{2}$, and any ordered sequence of $q = o(1.3803^{\frac{s}{2}})$ forward/inverse queries, with associated round values $R[i,0], \ldots, R[i,k+1]$ for $i = 1 \ldots q$, if the 5-XOR condition does not hold for this sequence of queries then there is no r^{th} round value collision for these queries, for all $r \in \{s \ldots (k-s)\}$.*

Note that is is simply a structural property of the k-round LR construction that holds irrespective of the round functions used in the construction. We will provide a (very) high level overview of the proof followed by the formal proof. The proof essentially considers starts by assuming that the *5-XOR condition* does not hold for the given sequence of queries in which two different queries collide in the r^{th} round value. Then we prove the existence of an exponential number of queries in this sequence as follows:

1. We first show that each round value (generated before the r^{th} round value) in the later of the two colliding queries collides with the corresponding round value in an earlier query.

2. Next we prove that if each of these colliding queries (made earlier) are different and could only have been made in an order such that at least half of these are in strict ascending/descending order, in terms of the order in which they were made.

3. Then we show that for (almost) each of the queries in this strict ascending/descending sequence, there exists another strict ascending (or descending) sequence of queries (each of which is different from the ones already considered).

4. Finally, we note that this argument can be continued recursively, without double counting, and we get a recursion for the number of queries. Upon solving this recursion, we get that the number of queries in this sequence is $\Omega(1.3803^{s/2})$.

Proof: Assume that the 5-XOR condition does not hold for the given sequence of queries. Without loss of generality, say one of the queries involved in the r^{th} (where $r \in \{s \ldots (k-s)\}$, for $s \leq (k/2)$) round value collision is the last (q^{th}) query. If this were not the case then we can disregard all queries following the colliding query that was made later, and argue on the smaller sequence of queries that remains. In addition, we also assume that the given sequence of queries does not consist of duplicate queries. If not then we can disregard all but the first of these identical queries. This will not weaken our conclusion, but makes our argument easier to explain.

We represent the j^{th} round value associated with the i^{th} query as $R[i,j]$. Thus we know that $\exists i < q \ : \ R[q,r] = R[i,r]$. We maintain a q vector b that denotes the direction of each query. Thus $\mathsf{b}[i] = 1$ denotes that the i^{th} query is a forward query, while $\mathsf{b}[i] = 0$ denotes that it is an inverse query. We define a "first occurrence" query function for each round value, i.e. $\mathsf{p} : \{1 \dots q\} \times \{0 \dots k+1\} \to \{1 \dots q\}$. For any round value $R[i,j]$, $\mathsf{p}(i,j)$ is the *least input number* such that $R[\mathsf{p}(i,j),j] = R[i,j]$.

If the colliding round number $r \leq k/2$, then we get a worse lower bound if the q^{th} query is a forward query. Otherwise, we get a worse lower bound if it is an inverse query. Since the two cases are symmetrical, we will only describe here the argument when $r \leq k/2$ and assuming that the q^{th} query is a forward query.

As the first step of our argument, we prove that all the round values $R[q,1] \dots R[q,r-1]$ collide with the corresponding round value in an earlier query.

Claim 19. *If $\exists i < q \ : \ R[q,r] = R[i,r]$, then each of the round values $R[q,1] \dots R[q,r-1]$ were defined before the q^{th} query was made. That is,*

$$\forall j \in \{1 \dots r\} \ : \ \mathsf{p}(q,j) < q$$

proof of claim 19: We will use induction on the round number j to show that $\mathsf{p}(q, j) < q$. We start the induction with $j = r$ and go down to $j = 1$. For $j = r$, we already know that $\mathsf{p}(q, r) = i$ from the statement of the claim.

Now say the same holds for all $j = r \dots c$ (for $c \leq r$), then we will show that the $(c-1)^{th}$ round value also collides with the corresponding round value in an earlier query. Say, for the sake of contradiction, that $R[q, c-1]$ is a new round value in input number q (i.e. $\mathsf{p}(q, (c-1)) = q$). Then the round function evaluation $f_{(c-1)}\left(R[q, c-1]\right)$ is a new round function evaluation, generating the new round value $R[q, c]$. But $R[q, c] = R[\mathsf{p}(q, c), c]$, and $\mathsf{p}(q, c) < q$ by induction hypothesis. This contradicts the fact that the 5-XOR condition does not hold for the given sequence of queries. Thus, $\mathsf{p}(q, j) < q$ for all $j = 1 \dots r$. $\square$

Hence all the round values $R[q, 1] \dots R\left[q, r\right]$ already occur before query number q. As our next step, we will show that the order in which the queries $\mathsf{p}(q, 1) \dots \mathsf{p}\left(q, r\right)$ are made could be one of very few possible orders.

Claim 20. *There is a round number $j \in \{1 \dots r\}$, such that,*

$$\begin{array}{ccccccccc} \mathsf{p}(q,1) & > & \dots & > & \mathsf{p}(q,j) & & & & \\ & & & & \mathsf{p}(q,j) & < & \dots & < & \mathsf{p}\left(q,r\right) \end{array}$$

That is, the round value $R[q, j]$ was defined before any of the other round values $R[q, 1] \dots R\left[q, r\right]$. Moreover, the queries $\mathsf{p}(q, j) \dots \mathsf{p}\left(q, r\right)$ were made in this order, while the queries $\mathsf{p}(q, 1) \dots \mathsf{p}(q, j)$ were made in the reverse *order.*

proof of claim 20: We will first prove that for any three consecutive round values $R[q,(i-1)], R[q,i]$ and $R[q,(i+1)]$ (where $i \in \{2 \ldots r-1\}$), it holds that,

$$[\mathsf{p}(q,(i-1)) > \mathsf{p}(q,i)] \vee [\mathsf{p}(q,i) < \mathsf{p}(q,(i+1))]$$

The claim will then follow as a straightforward consequence.

Assume to the contrary that $\mathsf{p}(q,(i-1)) \leq \mathsf{p}(q,i)$ and $\mathsf{p}(q,(i+1)) \leq \mathsf{p}(q,i)$ for some $i \in \{2, r-1\}$. If $\mathsf{p}(q,(i-1)) = \mathsf{p}(q,i)$ (or $\mathsf{p}(q,i) = \mathsf{p}(q,(i+1))$) then it is easy to verify that queries $\mathsf{p}(q,i)$ and q are the same, which is not the case by assumption. Thus, we have the case that $\mathsf{p}(q,(i-1)) < \mathsf{p}(q,i)$ and $\mathsf{p}(q,i) > \mathsf{p}(q,(i+1))$. But we know from the design of Ψ_k that,

$$f_i(R[\mathsf{p}(q,i),i]) = R[\mathsf{p}(q,i),(i-1)] \oplus R[\mathsf{p}(q,i),(i+1)]$$

It is also the case that,

$$\begin{array}{rlcl}
 & f_i(R[q,i]) & = & R[q,(i-1)] \oplus R[q,(i+1)] \\
\Rightarrow & f_i(R[\mathsf{p}(q,i),i]) & = & \begin{array}{l} R[\mathsf{p}(q,(i-1)),(i-1)] \\ \oplus R[\mathsf{p}(q,(i+1)),(i+1)] \end{array} \\
\Rightarrow & \begin{array}{l} R[\mathsf{p}(q,i),(i-1)] \\ \oplus R[\mathsf{p}(q,i),(i+1)] \end{array} & = & \begin{array}{l} R[\mathsf{p}(q,(i-1)),(i-1)] \\ \oplus R[\mathsf{p}(q,(i+1)),(i+1)] \end{array}
\end{array}$$

Thus, if $\mathsf{b}[\mathsf{p}(q,i)] = 0$ then $R[\mathsf{p}(q,i),(i-1)]$ can be represented as an XOR of three previously existing round values otherwise $R[\mathsf{p}(q,i),(i+1)]$ has such an XOR representation. In any case, this will give a 5-XOR condition which we know does not hold. Thus we can say that

$$\forall i \in \{2 \ldots r-1\} \;\;:\;\; [\mathsf{p}(q,(i-1)) > \mathsf{p}(q,i)] \vee [\mathsf{p}(q,i) < \mathsf{p}(q,(i+1))]$$

Now it is a straightforward task to verify that the query orders consistent with this constraint are exactly the ones in the statement of claim 20. □

From claim 20, we can deduce that there exist at least $\frac{r}{2}$ consecutive round values in the q^{th} query, whose "first occurrence" queries are in strictly ascending/descending temporal order. Since $r < \frac{k}{2}$, we will get a worse lower bound on the number of queries if we assume that $q > \mathsf{p}(q,1) > \ldots > \mathsf{p}\left(q, \frac{r}{2}\right)$. If on the other hand, $q > \mathsf{p}\left(q,r\right) > \ldots > \mathsf{p}\left(q, \frac{r}{2}\right)$ we can show that $q = \Omega(1.3803^r)$. Thus, we assume that $q > \mathsf{p}(q,1) > \ldots > \mathsf{p}\left(q, \frac{r}{2}\right)$.

As our next step, we will prove a general property of such a strictly ordered sequence of "first occurrence" queries of consecutive round values. For this purpose, consider any three consecutive "first occurrence" queries out of such a sequence, say $i_j = \mathsf{p}(\ell, j)$, $i_{j+1} = \mathsf{p}(\ell,(j+1))$ and $i_{j+2} = \mathsf{p}(\ell,(j+2))$ such that $i_j > i_{j+1} > i_{j+2}$. We will determine the order in which the "first occurrence" queries of the round values of the i_j^{th} query could have been made in this case. Additionally, we will also determine the order of these queries relative to the queries i_j, i_{j+1} and i_{j+2}.

We essentially show that if the i_j^{th} query is a forward query then the round values $R[i_j, 1] \ldots R[i_j, (j-1)]$ collide with corresponding round values in some query before the i_j^{th} query. Moreover, we also show that the queries $\mathsf{p}(i_j, 1) \ldots \mathsf{p}(i_j, j-2)$ were made after the i_{j+1}^{th} query, but before the i_j^{th} query. If the i_j^{th} query is an inverse query, then we prove the same conditions for the round values $R[i_j, (j+1)] \ldots R[i_j, k]$. This is formally stated in the following claim.

Claim 21. *Let the queries numbered i_j, i_{j+1} and i_{j+2} be the "first occurrence" queries of the round values $R[\ell, j]$,$R[\ell, j+1]$ and $R[\ell, j+2]$, respectively. Moreover, say that $i_j > i_{j+1} > i_{j+2}$. If the i_j^{th} query is a forward query (i.e. $\mathsf{b}[i_j] = 1$) then,*

$$i_j > \mathsf{p}(i_j, 1) > \ldots > \mathsf{p}(i_j, j-2) > i_{j+1}$$

On the other hand, if $\mathsf{b}[i_j] = 0$ then,

$$i_j > \mathsf{p}(i_j, k) > \ldots > \mathsf{p}(i_j, j+2) > i_{j+1}$$

proof of claim 21: Let us start by considering the case that $\mathsf{b}[i_j] = 1$. In this case, we analyze the round values $R[i_j, 1] \ldots R[i_j, (j-1)]$. Consider the round value $R[i_j, (j-1)]$. If this round value does not collide with a corresponding round value before the i_j^{th} query, then $f_{j-1}(R[i_j, (j-1)])$ is a new round function evaluation and $R[i_j, j]$ is the newly generated round value. But we know that

$$\begin{aligned} & f_{j+1}(R[\ell,(j+1)]) &=& \; R[\ell,j]\oplus R[\ell,(j+2)] \\ \Rightarrow\; & f_{j+1}(R[i_{j+1},(j+1)]) &=& \; R[i_j,j]\oplus R[i_{j+2},(j+2)] \\ \Rightarrow\; & R[i_j,j] &=& \; R[i_{j+2},(j+2)]\oplus R[i_{j+1},j]\oplus R[i_{j+1},(j+2)] \end{aligned}$$

And since $i_j > i_{j+1} > i_{j+2}$, this will give a representation of the newly generated round value $R[i_j,j]$ in terms of 3 previously existing round values, which violates the fact that the 5-XOR condition does not hold. Thus, we can deduce that $\mathsf{p}(i_j,j-1) < i_j$. Now we can argue inductively (similar to claims 19) and show that each of the round values $R[i_j,1]\ldots R[i_j,(j-2)]$ collide with corresponding round values in earlier queries as well.

<u>Conclusion 1:</u> We can deduce that $\forall j' \in \{1\ldots j-1\}\;:\;\mathsf{p}(i_j,j') < i_j$.

Now we will try to find the order in which these queries $\mathsf{p}(i_j,j')$ could have been made. In addition, since we know that $i_{j+2} < i_{j+1} < i_j$, we will also be interested in the order of the queries $\mathsf{p}(i_j,j')$ relative to the i_{j+1}^{th} and i_{j+2}^{th} queries. Let us start by concentrating our attention on the queries $i_{j+1}, \mathsf{p}(i_j,(j-1))$ and $\mathsf{p}(i_j,(j-2))$.

Consider the case that $\mathsf{p}(i_j,(j-1)) < i_{j+1}$ and $\mathsf{p}(i_j,(j-2)) < i_{j+1}$. Then we can deduce that,

$$\begin{aligned}
& f_{j+1}(R[i_{j+1},(j+1)]) && = && R[i_j,j] \oplus R[i_{j+2},(j+2)] \\
\Rightarrow\ & f_{j+1}(R[i_{j+1},(j+1)]) && = && \begin{array}{l} R[i_{j+2},(j+2)] \oplus R[i_j,(j-2)] \\ \oplus f_{j-1}(R[i_j,(j-1)]) \end{array} \\
\Rightarrow\ & \begin{array}{l} R[i_{j+1},(j+2)] \\ \oplus R[i_{j+1},j] \end{array} && = && \begin{array}{l} R[i_{j+2},(j+2)] \oplus R[\mathsf{p}(i_j,(j-2)),(j-2)] \\ \oplus R[\mathsf{p}(i_j,(j-1)),j] \oplus R[\mathsf{p}(i_j,(j-1)),(j-2)] \end{array}
\end{aligned}$$

Thus depending on whether i_{j+1} is a forward or inverse query, we get a representation of $R[i_{j+1},j]$ or $R[i_{j+1},(j+2)]$ as an XOR of five previous round values and since $R[i_{j+1},(j+1)]$ is a new round value this contradicts the fact that 5-XOR condition does not hold for the given sequence of queries. Thus we can deduce that,

$$\mathsf{p}(i_j,(j-1)) > i_{j+1} \text{ or } \mathsf{p}(i_j,(j-2)) > i_{j+1} \tag{5.1}$$

Next we consider the case that $\mathsf{p}(i_j,(j-2)) < \mathsf{p}(i_j,(j-1))$ as well as $i_{j+1} < \mathsf{p}(i_j,(j-1))$. In this case, we observe that,

$$\begin{aligned}
& f_{j-1}(R[i_j,(j-1)]) && = && R[i_j,(j-2)] \oplus R[i_j,j] \\
\Rightarrow\ & f_{j-1}(R[\mathsf{p}(i_j,(j-1)),(j-1)]) && = && \begin{array}{l} R[i_j,(j-2)] \oplus f_{j+1}(R[i_{j+1},(j+1)]) \\ \oplus R[i_{j+2},(j+2)] \end{array} \\
\Rightarrow\ & \begin{array}{l} R[\mathsf{p}(i_j,(j-1)),(j-2)] \\ \oplus R[\mathsf{p}(i_j,(j-1)),j] \end{array} && = && \begin{array}{l} R[\mathsf{p}(i_j,(j-2)),(j-2)] \oplus R[i_{j+1},j] \\ \oplus R[i_{j+1},(j+2)] \oplus R[i_{j+2},(j+2)] \end{array}
\end{aligned}$$

Now depending on whether the $\mathsf{p}(i_j,(j-1))^{th}$ query is a forward or inverse query, we can derive a 5 XOR representation of either $R[\mathsf{p}(i_j,(j-1)),(j-2)]$ or $R[\mathsf{p}(i_j,(j-1)),j]$ in terms of previously existing round values, thus violating the fact that the 5-XOR condition does not hold. Hence we deduce that

$$\mathsf{p}(i_j,(j-2)) > \mathsf{p}(i_j,(j-1)) \text{ or } i_{j+1} > \mathsf{p}(i_j,(j-1)) \tag{5.2}$$

In order to satisfy both equations 5.1 and 5.2, we need that $\mathsf{p}(i_j,(j-2)) > \mathsf{p}(i_j,(j-1))$ as well as $\mathsf{p}(i_j,(j-2)) > i_{j+1}$.

<u>Conclusion 2:</u> We can deduce that the only two possible orders for these three queries are

$$\mathsf{p}(i_j,(j-2)) > \mathsf{p}(i_j,(j-1)) > i_{j+1} \text{ or } \mathsf{p}(i_j,(j-2)) > i_{j+1} > \mathsf{p}(i_j,(j-1))$$

In either case, we can deduce from *conclusion 2* that $\mathsf{p}(i_j,(j-2)) > \mathsf{p}(i_j,(j-1))$. Next consider the query $\mathsf{p}(i_j,(j-3))$. If $\mathsf{p}(i_j,(j-2)) > \mathsf{p}(i_j,(j-3))$ as well, then we can deduce that

$$\begin{array}{lcl} R[\mathsf{p}(i_j,(j-2)),(j-3)] & = & R[\mathsf{p}(i_j,(j-1)),(j-1)] \\ \oplus R[\mathsf{p}(i_j,(j-2)),(j-1)] & & \oplus R[\mathsf{p}(i_j,(j-3)),(j-3)] \end{array}$$

This will give a representation of either $R[\mathsf{p}(i_j,(j-2)),(j-1)]$ or $R[\mathsf{p}(i_j,(j-2)),(j-3)]$ in terms of 3 previously existing round values depending on whether the $\mathsf{p}(i_j,(j-2))^{th}$ query is a forward or an inverse query, respectively.

In either case, this violates the fact that the 5-XOR condition does not hold for the given sequence of queries. Thus, we can deduce that $\mathsf{p}(i_j, (j-3)) > \mathsf{p}(i_j, (j-2)) > \mathsf{p}(i_j, (j-1))$. Now this same argument can be continued and using *conclusion 2*, we can prove that

$$i_j > \mathsf{p}(i_j, 1) > \ldots > \mathsf{p}(i_j, (j-2)) > i_{j+1}$$

If the query number i_j is an inverse query, then we can carry out a symmetric argument by considering the round values $R[i_j, (j+1)] \ldots R[i_j, k]$ instead of $R[i_j, (j-1)] \ldots R[i_j, 1]$. Then it can be deduced that

$$i_j > \mathsf{p}(i_j, k) > \ldots > \mathsf{p}(i_j, (j+2)) > i_{j+1}$$

$\square$

We will apply claim 21 to the sequence of first occurrence queries $\mathsf{p}(q,1) > \ldots > \mathsf{p}\left(q, \frac{r}{2}\right)$. Thus consider any $j = 1 \ldots \frac{r}{2} - 2$, and the first occurrence queries $\mathsf{p}(q,j), \mathsf{p}(q,(j+1))$ and $\mathsf{p}(q,(j+2))$. Since $j \leq \frac{r}{2} \leq \frac{k}{4}$, we will get a worse bound if the $\mathsf{p}(q,j)^{th}$ query is a forward query. In particular, we can use claim 21 to show in this case that the round values $R[\mathsf{p}(q,j), 1] \ldots R[\mathsf{p}(q,j), (j-1)]$ collide with corresponding round values in earlier queries. On the other hand, if this is an inverse query then we can show that each of the round values $R[\mathsf{p}(q,j), (j+1)] \ldots R[\mathsf{p}(q,j), k]$ collide with corresponding round values in an earlier query. The former case clearly gives us a worse lower bound on the number of queries, and hence we assume that the $\mathsf{p}(q,j)^{th}$ query is an inverse query.

Hence considering query $\mathsf{p}(q,j)$, we can deduce that at least another $(j-2)$ queries we made before it but after the query $\mathsf{p}(q,(j+1))$. We can also deduce from claim 21 that these $(j-2)$ "first occurrence" queries are also made in strictly descending temporal order, that is

$$\mathsf{p}(\mathsf{p}(q,j),1) > \ldots > \mathsf{p}(\mathsf{p}(q,j),(j-2))$$

Since each of these sequence of queries is in strict temporal order, we can apply claim 21 to each of these sequence of queries and continue in this recursive fashion. Moreover, we also show that any queries that we count at a certain level of the recursion in this fashion lies strictly in between two consecutive queries from the previous level, we can deduce that we do not perform any double counting. In figure 5.2, we illustrate an example of a query tree with the first three levels indicated. Here the two queries that collide in the $(k/2)^{th}$ round value are indicated as "colliding queries" 1 and 2, in the order they were made.

In order to bound from below the number of queries q that produce a collision on the r^{th} round value, we will need to count the number of queries that are bound to exist by the argument above. Let $\mathcal{Q}(j)$ be a recursively defined variable that denotes the minimum number of queries ℓ needed to get round values $R[\ell,1]\ldots R[\ell,j]$ with their first occurrence queries made in the

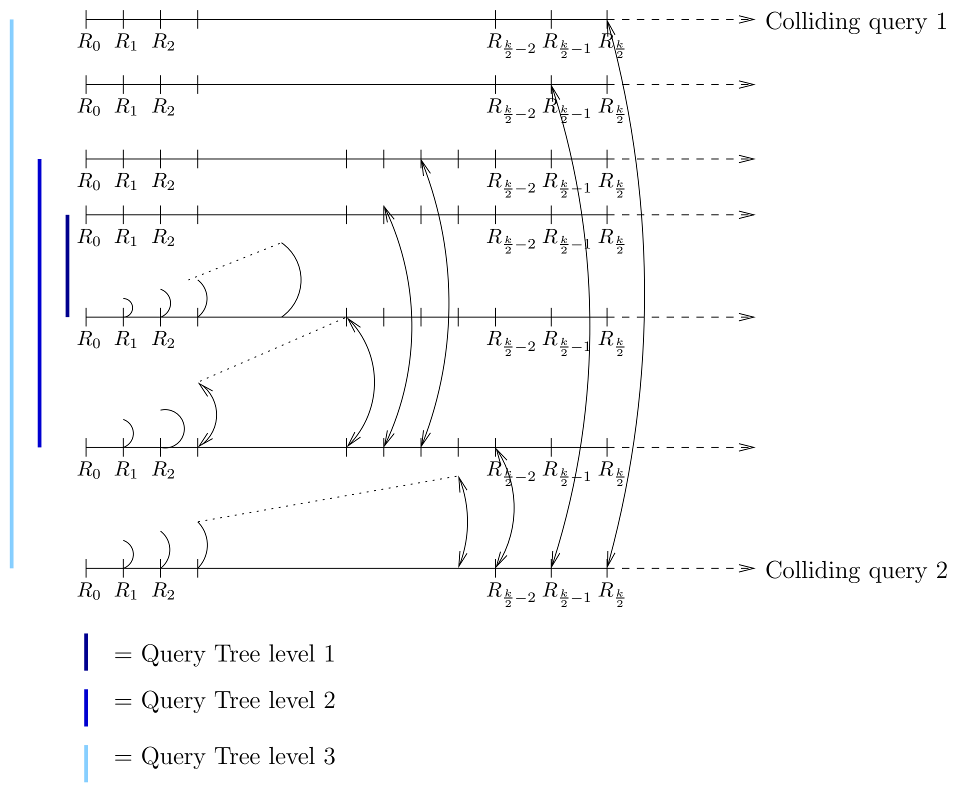


Figure 5.2: Example of the first three levels of a query tree. All the queries in this example are assumed to be forward queries.

order $\mathsf{p}(i,1) > \ldots > \mathsf{p}(i,j)$. Using claim 21, we get the following expression for $\mathcal{Q}(j)$,

$$\begin{aligned}
\mathcal{Q}(j) &= j + \textstyle\sum_{\ell=3}^{j-2} \mathcal{Q}(\ell-2) \\
\Rightarrow \quad \mathcal{Q}(j) &= \mathcal{Q}(j-1) + \mathcal{Q}(j-4) + 1 \\
\Rightarrow \quad \mathcal{Q}(j) &= 2\cdot\mathcal{Q}(j-1) - \mathcal{Q}(j-2) + \mathcal{Q}(j-4) - \mathcal{Q}(j-5)
\end{aligned}$$

The solution to the above homogeneous recurrence equation can be expressed in terms of the powers of the roots of the following algebraic equation:

$$x^5 - 2x^4 + x^3 - x^4 + 1 = 0$$

This equation has only one root greater than 1, which is 1.3803. Thus we can represent the solution of the above recurrence as:

$$\mathcal{Q}(j) = \Theta(1.3803^j)$$

From claim 20, we get that if any query collides with an earlier query in the r^{th} round value, we can find a strictly increasing/decreasing sequence of $\frac{r}{2}$ "first occurrence" queries. Thus, we get that

$$\begin{array}{rcll} & q & \geq & \mathcal{Q}\left(\frac{r}{2}\right) \\ \Rightarrow & q & = & \Omega\left(1.3803^{r/2}\right) \\ \Rightarrow & q & \geq & \Omega\left(1.3803^{s/2}\right) \text{ , since } r \in \{s \ldots (k-s)\} \end{array}$$

The above proof only took into account the case that $r < k/2$. If $r > k/2$ then a similar argument can be carried out by swapping forward queries with inverse queries and we can derive that $q = \Omega\left(1.3803^{(k-r)/2}\right)$. In either case, we get the bound that $q = \Omega\left(1.3803^{s/2}\right)$, since $r \in \{s \ldots (k-s)\}$ and $s \leq (k/2)$. $\square$

5.3.2 A Slightly Weaker Result

Next we state a more restricted version of the combinatorial lemma, when the adversary only makes forward queries to the Feistel construction. This lemma will be useful in certain scenarios when the attacker only makes forward queries to the Feistel-based construction (for instance, in *domain extension of MACs* in next chapter).

Lemma 21. *Let Ψ_k be a k-round LR construction that uses fixed and arbitrary round functions $f_1 \dots f_k$. For any round number s, and any ordered sequence of $q = o(1.3803^{\frac{s}{2}})$ forward queries, with associated round values $R[i,0], \dots, R[i,k+1]$ for $i = 1 \dots q$, if the 5-XOR condition does not hold for this sequence of forward queries then there is no r^{th} round value collision for these queries, for all $r \geq s$.*

The proof of this lemma is similar to that of lemma 20, but is slightly simpler since the query sequence only consists of forward queries. We provide it next for completeness.

Proof: We start by assuming that the 5-XOR condition does not hold for the given sequence of forward queries. Without loss of generality, say one of the queries involved in the r^{th} round collision (for $r \geq s$) is the last query, i.e. the q^{th} query. If this is not the case, then we can easily ignore the queries following the collision queries and get a smaller sequence of queries for which the property holds. We also assume that the given sequence of queries does not consist of duplicate queries. If this is not the case then we can ignore all but the first one of all such identical queries and it will not weaken our conclusion, while making the argument easier to describe.

We represent the j^{th} round value associated with the i^{th} query as $R[i,j]$. Thus we know that $\exists i < q \;\; : \;\; R\left[q,r\right] = R\left[i,r\right]$. We define a "first occurrence" function for each round value, i.e. $\mathsf{p} : \{1 \ldots q\} \times \{0 \ldots k+1\} \rightarrow \{1 \ldots q\}$. For any round value $R[i,j]$, $\mathsf{p}(i,j)$ is the *least input number* such that $R[\mathsf{p}(i,j),j] = R[i,j]$.

In the first step, we will prove that all the round values $R[q,1] \ldots R[q,(r-1)]$ collide with the corresponding round value in an earlier query. As a first step, we prove that all the round values $R[q,1] \ldots R\left[q,r-1\right]$ collide with the corresponding round value in an earlier query.

Claim 22. *If $\exists i < q \;\; : \;\; R\left[q,r\right] = R\left[i,r\right]$, then each of the round values $R[q,1] \ldots R\left[q,(r-1)\right]$ were already defined before the q^{th} query was made. That is,*

$$\forall j \in \{1 \ldots r\} \;\; : \;\; \mathsf{p}(q,j) < q$$

proof of claim 22: We will use induction on the round number j to show that $\mathsf{p}(q,j) < q$. We start the induction with $j = r$ and go down to $j = 1$. For $j = r$, we already know that $\mathsf{p}(q,r) = i$ from the statement of the claim.

Now say the same holds for all $j = r \ldots c$ (for $c \leq r$), then we will show that the $(c-1)^{th}$ round value also collides with the corresponding round value in an earlier query. Say, for the sake of contradiction, that $R[q,c-1]$ is a new round value in query number q (i.e. $\mathsf{p}(q,c-1) = q$). Then the round function evaluation $f_{(c-1)}\left(R[q,c-1]\right)$ is a new round function evaluation, and hence $R[q,c]$ is the new round function value generated as a result. But $R[q,c] = R[\mathsf{p}(q,c),c]$, and $\mathsf{p}(q,c) < q$ by the induction hypothesis, which is a 1-XOR representation of the new round value $R[q,c]$. This contradicts the fact that the 5-XOR condition does not hold for this sequence of queries. $\square$

Thus, we know that all the round values $R[q,1] \ldots R\left[q,r\right]$ were defined strictly before input number q. As our next step, we will show that the order in which the queries $\mathsf{p}(q,1) \ldots \mathsf{p}\left(q,r\right)$ are made can only be one of very few specific orders.

Claim 23. *There is a round number $j \in \{1 \ldots r\}$, such that,*

$$\begin{array}{ccccccccc} \mathsf{p}(q,1) & > & \ldots & > & \mathsf{p}(q,j) & & & & \\ & & & & \mathsf{p}(q,j) & < & \ldots & < & \mathsf{p}\left(q,r\right) \end{array}$$

That is, the queries $\mathsf{p}(q,j) \ldots \mathsf{p}\left(q,r\right)$ were made in this order, while the queries $\mathsf{p}(q,1) \ldots \mathsf{p}(q,j)$ were made in the reverse order.

proof of claim 23: We will first prove that for any three consecutive round values $R[q,(i-1)], R[q,i]$ and $R[q,(i+1)]$ (where $i \in \{2 \ldots r-1\}$), it holds that,

$$[\mathsf{p}(q,(i-1)) > \mathsf{p}(q,i)] \vee [\mathsf{p}(q,i) < \mathsf{p}(q,(i+1))]$$

The claim will then follow as a straightforward consequence.

Assume to the contrary that $\mathsf{p}(q,(i-1)) \leq \mathsf{p}(q,i)$ and $\mathsf{p}(q,(i+1)) \leq \mathsf{p}(q,i)$

for some $i \in \{2, r-1\}$. We can easily see that $\mathsf{p}(q,(i-1)) \neq \mathsf{p}(q,i)$ (and $\mathsf{p}(q,i) \neq \mathsf{p}(q,(i+1)))$ since otherwise the queries $\mathsf{p}(q,i)$ and q will be the same.

Thus, it must be the case that $\mathsf{p}(q,(i-1)) < \mathsf{p}(q,i)$ and $\mathsf{p}(q,i) > \mathsf{p}(q,(i+1))$. But we know from the design of Ψ_k that,

$$f_i(R[\mathsf{p}(q,i),i]) = R[\mathsf{p}(q,i),(i-1)] \oplus R[\mathsf{p}(q,i),(i+1)]$$

It is also the case that,

$$\begin{array}{rlcl} & f_i(R[q,i]) & = & R[q,(i-1)] \oplus R[q,(i+1)] \\ \Rightarrow & f_i(R[\mathsf{p}(q,i),i]) & = & R[\mathsf{p}(q,(i-1)),(i-1)] \oplus R[\mathsf{p}(q,(i+1)),(i+1)] \\ \Rightarrow & R[\mathsf{p}(q,i),(i+1)] & = & \begin{array}{l} R[\mathsf{p}(q,(i-1)),(i-1)] \oplus R[\mathsf{p}(q,(i+1)),(i+1)] \\ \oplus R[\mathsf{p}(q,i),(i-1)] \end{array} \end{array}$$

Thus the new round value $R[\mathsf{p}(q,i),(i+1)]$ can be represented as an XOR of 3 previously existing round values. This will give a 5-XOR condition which we know does not hold. Thus we can say that

$$\forall i \in \{2 \ldots r-1\} \;\; : \;\; [\mathsf{p}(q,(i-1)) > \mathsf{p}(q,i)] \vee [\mathsf{p}(q,i) < \mathsf{p}(q,(i+1))]$$

Now it is a straightforward task to verify that the query orders consistent with this constraint are exactly the ones in the statement of claim 23. □

From claim 23, we can deduce that there exist at least $\frac{r}{2}$ consecutive round values in the q^{th} query, whose "first occurrence" queries are in strictly ascending/descending temporal order. We note that for the given game, the worse case is if the queries $\mathsf{p}(q,1)\ldots\mathsf{p}\left(q,\frac{r}{2}\right)$ are made in the reverse order (i.e. this is the part with the greater number of strictly ordered queries). In fact, if it is the case that $\mathsf{p}\left(q,\frac{r}{2}\right) < \ldots < \mathsf{p}(q,r)$, then we can show that the number of queries $q = \mathcal{O}(1.3803^r)$. Since we wish to find the lower bound on the number of queries q, we assume that $q > \mathsf{p}(q,1) > \ldots > \mathsf{p}\left(q,\frac{r}{2}\right)$.

As our next step, we will prove a general property of such a strictly ordered sequence of "first occurrence" queries of consecutive round values. For this purpose, we consider the first occurrence queries $\mathsf{p}(\ell,j), \mathsf{p}(\ell,(j+1))$ and $\mathsf{p}(\ell,(j+2))$, denoted by i_j, i_{j+1} and i_{j+2} respectively. Assume that these queries are made in the order $i_j > i_{j+1} > i_{j+2}$. We will show that all the round values $R[i_j,1]\ldots R[i_j,j-1]$ collide with corresponding round values in queries before the i_j^{th} query. Moreover, we also show that the queries $\mathsf{p}(i_j,1)\ldots\mathsf{p}(i_j,j-2)$ were made after the i_{j+1}^{th} query, but before the i_j^{th} query. This is formally stated in the following claim.

Claim 24. *Let the queries numbered i_j, i_{j+1} and i_{j+2} be the "first occurrence" queries of the round values $R[\ell,j]$,$R[\ell,j+1]$ and $R[\ell,j+2]$ (respectively) for any query ℓ. Moreover, say that $i_j > i_{j+1} > i_{j+2}$. Then, it is the case that*

$$i_j > \mathsf{p}(i_j,1) > \ldots > \mathsf{p}(i_j,j-2) > i_{j+1}$$

proof of claim 24: Consider the round value $R[i_j, j-1]$. If this does not collide with a corresponding round value in an earlier query, then $f_{j-1}(R[i_j, j-1])$ is a new round function evaluation and $R[i_j, j]$ is the new round value generated as a result. Now we know that

$$\begin{aligned} & f_{j+1}(R[\ell, j+1]) && = && R[\ell, j] \oplus R[\ell, j+2] \\ \Rightarrow\ & f_{j+1}(R[i_{j+1}, (j+1)]) && = && R[i_j, j] \oplus R[i_{j+2}, j+2] \\ \Rightarrow\ & R[i_j, j] && = && R[i_{j+2}, (j+2)] \oplus R[i_{j+1}, j] \oplus R[i_{j+1}, (j+2)] \end{aligned}$$

Since $i_j > i_{j+1} > i_{j+2}$, this would give us a 5-XOR condition involving the new round value $R[i_j, j]$ which we know does not hold. Hence we know that $\mathsf{p}(i_j, j-1) < i_j$. Now we can use induction to show that all the round values $R[i_j, 1] \ldots R[i_j, (j-2)]$ also collide with a corresponding round value in an earlier query.

<u>**Conclusion 1:**</u> We have deduced that $\forall j' \in \{1 \ldots j-1\} \ : \ \mathsf{p}(i_j, j') < i_j$.

Now we will try to find the order in which these "first occurrence" queries could have been made. In addition, we will also be interested in establishing the order of these queries relative to the queries i_{j+1} and i_{j+2}. Let us start by concentrating our attention on the queries i_{j+1}, $\mathsf{p}(i_j, (j-1))$ and $\mathsf{p}(i_j, (j-2))$.

First, consider the case that $\mathsf{p}(i_j, (j-1)) < i_{j+1}$ and $\mathsf{p}(i_j, (j-2)) < i_{j+1}$.

Then we know that,

$$\begin{aligned}
f_{j+1}(R[i_{j+1},(j+1)]) &= R[i_j,j] \oplus R[i_{j+2},(j+2)] \\
\Rightarrow \; f_{j+1}(R[i_{j+1},(j+1)]) &= \begin{array}{l} R[i_{j+2},(j+2)] \oplus R[i_j,(j-2)] \\ \oplus f_{j-1}(R[i_j,(j-1)]) \end{array} \\
\Rightarrow \; \begin{array}{l} R[i_{j+1},(j+2)] \\ \oplus R[i_{j+1},j] \end{array} &= \begin{array}{l} R[i_{j+2},(j+2)] \oplus R[\mathsf{p}(i_j,(j-2)),(j-2)] \\ \oplus R[\mathsf{p}(i_j,(j-1)),j] \oplus R[\mathsf{p}(i_j,(j-1)),(j-2)] \end{array}
\end{aligned}$$

Since, $R[i_{j+1},(j+1)]$ was occurs for the first time in the i_{j+1}^{th} query, we get a representation of the newly generated round value $R[i_{j+1},(j+2)]$ in terms of 5 previously existing round values. This contradicts the fact that the 5-XOR condition does not hold for these queries. Thus we can deduce that,

$$\mathsf{p}(i_j,(j-1)) > i_{j+1} \text{ or } \mathsf{p}(i_j,(j-2)) > i_{j+1} \tag{5.3}$$

Along similar lines, consider the case that $\mathsf{p}(i_j,(j-2)) < \mathsf{p}(i_j,(j-1))$ as well as $i_{j+1} < \mathsf{p}(i_j,(j-1))$. In this case, we observe that,

$$\begin{aligned}
f_{j-1}(R[i_j,(j-1)]) &= R[i_j,(j-2)] \oplus R[i_j,j] \\
\Rightarrow \; f_{j-1}(R[\mathsf{p}(i_j,(j-1)),(j-1)]) &= \begin{array}{l} R[i_j,(j-2)] \oplus f_{j+1}(R[i_{j+1},(j+1)] \\ \oplus R[i_{j+2},(j+2)] \end{array} \\
\Rightarrow \; \begin{array}{l} R[\mathsf{p}(i_j,(j-1)),(j-2)] \\ \oplus R[\mathsf{p}(i_j,(j-1)),j] \end{array} &= \begin{array}{l} R[i_j,(j-2)] \oplus R[i_{j+1},j] \\ \oplus R[i_{j+1},(j+2)] \oplus R[i_{j+2},(j+2)] \end{array}
\end{aligned}$$

Here the round value $R[\mathsf{p}(i_j,(j-1)),(j-1)]$ occurs for the first time in the

$\mathsf{p}(i_j,(j-1))^{th}$ query. Thus the newly generated round value $R[\mathsf{p}(i_j,(j-1)),j]$ can be represented as an XOR of 5 previously existing round values. This again contradicts the fact that the 5-XOR condition does not hold for the given sequence of queries. Hence we deduce that,

$$\mathsf{p}(i_j,(j-2)) > \mathsf{p}(i_j,(j-1)) \text{ or } i_{j+1} > \mathsf{p}(i_j,(j-1)) \tag{5.4}$$

In order to satisfy both equations 5.3 and 5.4, it is required that $\mathsf{p}(i_j,(j-2)) > \mathsf{p}(i_j,(j-1))$ as well as $\mathsf{p}(i_j,(j-2)) > i_{j+1}$.

Conclusion 2: We can deduce that the only possible orders for these three queries are

$$\mathsf{p}(i_j,(j-2)) > \mathsf{p}(i_j,(j-1)) > i_{j+1} \text{ or } \mathsf{p}(i_j,(j-2)) > i_{j+1} > \mathsf{p}(i_j,(j-1))$$

In either case, we can deduce from *conclusion 2* that $\mathsf{p}(i_j,(j-2)) > \mathsf{p}(i_j,(j-1))$. Next consider the query $\mathsf{p}(i_j,(j-3))$. If $\mathsf{p}(i_j,(j-2)) > \mathsf{p}(i_j,(j-3))$ as well, then we can deduce that

$$\begin{array}{ccc} R[\mathsf{p}(i_j,(j-2)),(j-3)] & = & R[\mathsf{p}(i_j,(j-1)),(j-1)] \\ \oplus R[\mathsf{p}(i_j,(j-2)),(j-1)] & & \oplus R[\mathsf{p}(i_j,(j-3)),(j-3)] \end{array}$$

This will give a representation of either $R[\mathsf{p}(i_j,(j-2)),(j-1)]$ in terms of 3 previously existing round values. This violates the fact that the 5-XOR condition does not hold for the given sequence of queries. Thus, we can deduce that $\mathsf{p}(i_j, (j-3)) > \mathsf{p}(i_j, (j-2)) > \mathsf{p}(i_j, (j-1))$. Now this same argument can be continued and using *conclusion 2*, we can prove that

$$i_j > \mathsf{p}(i_j, 1) > \ldots > \mathsf{p}(i_j, (j-2)) > i_{j+1}$$

$\square$

Now we can apply claim 24 to the sequence of first occurrence queries $\mathsf{p}(q, 1) > \ldots > \mathsf{p}\left(q, \frac{r}{2}\right)$. Thus for each query $\mathsf{p}(q, i)$ (for $i = 1 \ldots \frac{r}{2} - 2$), we can deduce that all the queries $\mathsf{p}(\mathsf{p}(q, i), (i-2)) \ldots \mathsf{p}(\mathsf{p}(q, i), 1)$ were made in this order between queries $\mathsf{p}(q, (i+1))$ and $\mathsf{p}(q, i)$. And since these queries are made strictly in between two consecutive queries from the previous level (i.e. $\mathsf{p}(q, (i+1))$ and $\mathsf{p}(q, i)$ in this case), we can also deduce that each of the queries in these sequences is different from the queries $\mathsf{p}(q, 1) \ldots \mathsf{p}\left(q, \frac{r}{2}\right)$.

Claim 24 can be applied to any sequence of strictly ordered “first occurrence” queries of consecutive round values. In particular, we can apply this claim to any of the new strictly ordered sequence of queries whose existence we showed here. Hence we can continue this argument recursively to prove the existence of many more queries before the last one, i.e. the q^{th} query.

Now we can find a lower bound on the number of queries q required in order to force a r^{th} round collision. To find this lower bound, we introduce a recursively defined variable $\mathcal{Q}(j)$, that denotes the minimum number of queries ℓ required to force a round value collision for each of the round values $R[\ell, 1] \ldots R[\ell, j]$ with a corresponding round value in a query prior to the ℓ^{th} query. From the above argument, we can deduce that

$$\begin{array}{rcl} \mathcal{Q}(j) & = & j + \sum_{i=3}^{j-2} \mathcal{Q}(i-2) \\ \Rightarrow \quad \mathcal{Q}(j) & = & \mathcal{Q}(j-1) + \mathcal{Q}(j-4) + 1 \\ \Rightarrow \quad \mathcal{Q}(j) & = & 2 \cdot \mathcal{Q}(j-1) - \mathcal{Q}(j-2) + \mathcal{Q}(j-4) - \mathcal{Q}(j-5) \end{array}$$

The solution to the above homogeneous equation can be expressed in terms of the powers of the roots of the following algebraic equation:

$$x^5 - 2x^4 + x^3 - x^4 + 1 = 0$$

This equation has only one root greater than 1, which is 1.3803. Thus we can represent the solution of the above recurrence as:

$$\mathcal{Q}(j) = \Theta(1.3803^j)$$

From claim 23, we can deduce that if the r^{th} round value in the q^{th} query collides with a corresponding round value in an earlier query, then (we get the worst lower bound if) the “first occurrence” queries of the round values

$R\left[q, \frac{r}{2}\right] \ldots R[q,1]$ are made in this order. Hence, we get that

$$\begin{aligned} q &\geq \mathcal{Q}\left(\tfrac{r}{2}\right) \\ \Rightarrow \quad q &= \Omega\left(1.3803^{r/2}\right) \\ \Rightarrow \quad q &\geq \Omega\left(1.3803^{s/2}\right) \text{ , since } r \geq s \end{aligned}$$

$\square$

5.3.3 Relevance for Feistel Applications

Both lemma 20 and 21 will be useful for the applications described in the next chapter. In particular, we will be interested in using the LR construction with round functions that resist the 5-XOR condition, when any adaptive adversary makes a polynomial number of queries while having access to all intermediate round values. We will specify this as a property of the *function ensemble* from which the round functions are derived. Here the function ensemble $\{\mathcal{F}_{n(\lambda)}\}_{\lambda \in \mathbb{N}}$ is such that $\mathcal{F}_n$ is a distribution over length-preserving functions on n bits. A function ensemble is called a *5-XOR resistant function family* if the LR construction using independently sampled functions from this ensemble resists the 5-XOR condition when queried a polynomial number of times by any adaptive adversary. More formally,

Definition 17 (5-XOR resistant function family). *A function ensemble $\{\mathcal{F}_{k(\lambda),n(\lambda)}\}_{\lambda \in \mathbb{N}}$, that consists of length preserving functions on n bits, is a*

5-XOR resistant function family *if for any adversary A,*

$$\Pr\left[A \text{ 5-XOR condition in } (A \leftrightarrow \Psi_{f_1,\ldots,f_k})\,|f_1,\ldots,f_k \leftarrow \mathcal{F}_{k,n}\right] \leq \epsilon_{xor} = negl(\lambda)$$

Here the advantage ϵ_{xor} of the adversary A depends the security parameter λ. The running time of A, input length n and number of Feistel rounds k are all polynomial functions of λ.

By applying lemmas 20 and 21 to a LR construction using round functions independently sampled from a 5-XOR resistant function family, we can derive the following corollary.

Corollary 2. *Let Ψ_k be a k-round LR construction that uses round functions that are independently sampled from a 5-XOR resistant function family consisting of functions on n bits. For any adversary A that adaptively makes permutation queries to Ψ_k, while observing the intermediate round values, it holds that*

- *if A makes both forward/inverse queries, then for any round number $s \leq (k/2)$ with $s = \omega(\log \lambda)$,*

  $$\Pr\left[\exists\ r^{th} \text{ r.v. collision in } A \leftrightarrow \Psi_k \text{ for some } r \in \{s \ldots (k-s)\}\right] \leq \epsilon_{xor}$$

- *if A makes only forward queries, then for any round number $s = \omega(\log \lambda)$,*

  $$\Pr\left[\exists\ r^{th} \text{ r.v. collision in } A \leftrightarrow \Psi_k \text{ for some } r \in \{s \ldots k\}\right] \leq \epsilon_{xor}$$

Here the bound ϵ_{xor} denotes the maximum advantage of the XOR finding adversary that runs in time $\mathcal{O}(t_A + (q_A k)^5)$, where t_A is the running time of the adversary A and q_A denotes the number of queries made by it. Also, t_A, q_A and the input length n are all polynomial in λ.

The proof of this corollary is quite straightforward since the 5-XOR finding adversary simply runs the collision finding adversary, and performs a brute force search for a 5-XOR condition when it finds a round value collision. From lemma 20 (or 21), such a 5-XOR condition is guaranteed to exist. We will use this corollary in each of the results in the next chapter.

Chapter 6

Implications for Feistel-based Primitives

In the previous chapter, we studied a general combinatorial property of Feistel networks. We also briefly mentioned that this result proves useful in applications of Feistel networks where one of the following two assumptions are violated:

(a) *the round functions are (pseudo)random*; and

(b) *(At least some of) the intermediate round values appearing during the Feistel computation are secret*

We noted that if anyone of these assumptions do not hold then all previous results that used Feistel network fail. We then gave a general combinatorial result that could be applied to scenarios where one (or both) of the above assumptions do not hold. In this chapter, we use this combinatorial result to provide constructions of new primitives as well as new (and stronger) constructions of previously known primitives.

6.0.4 Summary of results

Strong(er) PRP from any PRF. Our result can be used to give a construction of Strong PRPs from any PRF, that remains secure even when intermediate PRF computations are leaked to the attacker. Earlier, we gave examples of scenarios where such a construction may make sense. For instance, one might imagine a smartcard implementing a block cipher via the Feistel network using a secure chip implementing a PRF. In this case, the attacker might be able to observe the communication between the smartcard and the chip. More realistically, when the round functions of the block cipher are not PRFs, the attacker might get a lot of information about the intermediate values anyway. Our result implies that with a super-logarithmic number of rounds, a Feistel based block cipher is secure in such scenarios as well.

(Strong) unpredictable permutation (UP) from unpredictable functions (UF). We show that using a super-logarithmic number of Feistel rounds, one can construct (strong) unpredictable permutations using unpredictable functions in each round. Strong UPs are similar to UFs in the sense that no attacker should be able to predict an input-output pair which it has not explicitly queried the UP on. However, unlike UFs, here the attacker may also make inverse queries to the UP. Note that in this case, the round functions are not (pseudo)random and this assumption (a) is violated. However, we show that for unpredictable permutations, even assumption (b) may not be true. In particular, we give examples of (secure) UFs which when used as round functions in the Feistel network leak all intermediate round values to the attacker. Although artificial, this example illustrates that weaker round functions may no longer guarantee the secrecy of round values.

Verifiable Random Permutations. We apply our result to the problem of constructing *verifiable random permutations* (VRPs) from *verifiable random functions* (VRFs). VRFs and VRPs are verifiable analogs of PRFs and PRPs, respectively. Let us concentrate on VRFs first. Intuitively, regular PRFs have a limitation that one must trust the owner of the secret key that a given PRF value is correctly computed. And even when done so, a party receiving a correct PRF value cannot later convince some other party that the value is indeed correct (i.e., PRF values are "non-transferable"). In fact, since the function values are supposed to be (pseudo)random, it seems that such verifiability of outputs of a PRF would contradict its pseudorandomness. The way out of this contradiction was provided by Micali, Rabin and Vadhan [55], who introduced the notion of a VRF. Unlike PRFs, a VRF owner must be able to provide a short proof that any given VRF output is computed correctly. This implies that the VRF owner must publish a public key allowing others to verify the validity of such proofs. However, every "unopened" VRF value (i.e., one for which no proof was given yet) should still look indistinguishable from random, even if many other values were "opened" (by giving their proofs). Additionally, the public key should commit the owner of the VRF to all its function values in a unique way, even if the owner tries to select an "improper" public key. Micali et al. [55] also gave a secure construction of a VRF based on the RSA assumption. Since then several more efficient constructions of VRFs have been proposed based on various cryptographic assumptions; see [49, 24, 28].

The notion of a VRP, introduced in [27], naturally adds verifiability to PRPs, in exactly the same natural way as VRFs do to PRFs. We will describe some applications of VRPs later in this chapter. On the one hand, it is easy to see that a VRP (on a "non-trivial domain") is also a VRF, just like in the PRF/PRP case. On a first look, we might hope that the converse implication holds as well, by simply applying the Luby-Rackoff result to VRFs in place of PRFs. However, a moment of reflection shows that this is not the case. Indeed, the proof for the iterated Feistel construction *must include all the VRF values for the intermediate rounds*, together with their proofs. Thus, the attacker can legally obtain all the intermediate round values for every input/output that he queries, except for the one on which he is being "challenged". Thus rules out LR-type proof for this application.

We use our combinatorial result for the LR construction, to show that if a super-logarithmic number of Feistel rounds are used, then we get a secure *verifiable random permutation* from any *verifiable random function.*

Verifiable Unpredictable Permutations. We also consider the natural combination of the scenarios considered so far, exemplified by the task of constructing *verifiable unpredictable permutations* (VUPs) from *verifiable unpredictable functions* (VUFs) [55] (also called *unique signature schemes* [38, 49]). A VUF is defined in essentially the same way as VRFs, except that the pseudorandomness requirement for VRFs is replaced by a weaker unpredictability requirement. Similarly, VUPs, introduced in [27], are either the permutation analogs of VUFs, or, alternatively, unpredictable analogs of VRPs. Of course, as a VRP is also a VUP, we could attempt to build a VUP by actually building a VRP via the Feistel construction applied to a VRF, as suggested above. However, this seems quite wasteful since VUFs appear to be much easier to construct than VRFs. Indeed, although in theory VUFs are equivalent to VRFs [55], the "Goldreich-Levin-type" reduction from VUFs to VRFs in [55] is *extremely* inefficient (it loses exponential security and forces the authors to combine it with another inefficient tree construction). Moreover, several previous papers [55, 49] construct *efficient* VUFs based on relatively standard *computational* assumptions, while all the *efficient* VRF constructions [24, 28] are based on very ad hoc *decisional* assumptions. Thus, it is natural to study the security of the Feistel network when applied to VUFs. In this case, not only the round functions cannot be assumed pseudorandom, but also all the intermediate round values must be leaked together with their proofs of correctness, making this setting the most challenging to analyze. Using our result, we show that a super-logarithmic number of Feistel rounds with any secure VUFs gives a secure VUP.

Applications. In section 6.3, we illustrate many applications of our results, such as:

- We show how our results provide a "closer-to-reality" justification for the number of Feistel rounds heuristically used in practical block cipher constructions.

- Using our results, we provide the most efficient domain extension technique for length-preserving MACs without introducing any new assumptions.

- We show that VRPs immediately yield *non-interactive, setup-free, perfectly-binding commitment schemes.*

- VRPs can be used to fix a subtle security flaw in the non-interactive lottery system of Micali-Rivest [56].

- We show that these primitives can also be used to implement so called "invariant signatures" needed by Goldwasser and Ostrovsky [38].

- Other applications of VRPs, such as verifiable CBC encryption/decryption, verifiable huge (pseudo)random objects [36] or a "proof-transferable" implementation of the Ideal Cipher Model using a semi-trusted third party.

6.1 Preliminaries

In this section, we provide definitions for the cryptographic primitives that we will use throughout this chapter. We start by giving an alternative definition for *pseudorandom functions/permutations* that is different from the ones given in chapter 2 and is more suited for our results in this chapter. We will give these definitions for a function ensemble $\{H_\lambda\}_{\lambda\in\mathbb{N}}$ that is defined over the sequence of input/output sets $\{\{0,1\}^{a(\lambda)}, \{0,1\}^{b(\lambda)}\}_{\lambda\in\mathbb{N}}$. We will assume that the *key generating algorithm* $I(\lambda)$ outputs a bit string $s \in \{0,1\}^{c(\lambda)}$, where $c(\lambda)$ is the key length, and the keyed function will be represented as $H_s(\cdot)$. In this chapter, we will use the terms function ensemble and function family, interchangeably.

Definition 18 (Pseudorandom Functions). *A function ensemble $\{F_\lambda\}_{\lambda\in\mathbb{N}}$ is a pseudorandom function ensemble if for any probabilistic polynomial time (PPT) attacker pair $A = (A_1, A_2)$, which do not query their oracles on the challenge query, it holds that:*

$$\left|\Pr\left[b = b' \;\middle|\; \begin{array}{c} s \leftarrow I(\lambda); (x,\alpha) \leftarrow A_1^{F_s}(1^\lambda); y_0 \leftarrow F_s(x); \\ y_1 \leftarrow \{0,1\}^{b(\lambda)}; b \leftarrow \{0,1\}; b' \leftarrow A_2^{F_s}(y_b, \alpha) \end{array}\right] - \frac{1}{2}\right| = negl(\lambda)$$

We note that ,in asymptotic terms, definition 18 is equivalent to the definition 3 in chapter 2. We state this in the following lemma and provide a brief justification for the same.

Lemma 22. *Definitions 18 and 3 are equivalent definitions of PRFs (mod-*

ulo exact security). In particular, a (t, q, ϵ)-PRF according to definition 3, such that $\epsilon(\lambda)$ is negligible for any polynomials $t(\lambda)$ and $q(\lambda)$, is also a PRF according to definition 18, and vice versa.

Proof. We first show that a PRF according to definition 3 is also a PRF according to the definition 3 above. If this is not the case, then there is a PPT distinguisher D that has non-negligible advantage ϵ in distinguishing the uniform function ensemble and the PRF ensemble. If the D makes q oracle queries, then we consider $(q+1)$ hybrid scenarios. In the first hybrid, all oracle queries are responded to using the uniform random function. And in the i^{th} hybrid, the first $(i-1)$ queries are responded to using the PRF, while all remaining $(q-i+1)$ queries are responded to using the uniform random function. Thus, in the last hybrid, all queries are responded to using the PRF. Since the advantage ϵ of D in distinguishing between the first and $(q+1)^{th}$ hybrid is non-negligible. We can deduce that there is a $i \in \{1 \dots q\}$, such that D has an advantage of at least $\epsilon/(q+1)$ in distinguishing between the i^{th} and $(i+1)^{th}$ hybrid.

The attacker $A = (A_1, A_2)$ that we design essentially chooses a random $i \in \{0 \dots (q-1)\}$, and simulates the distinguisher as follows: The attacker A_1 simply runs the distinguisher D by responding to its oracle queries using its PRF oracle until the i^{th} query. Then it chooses the $(i+1)^{th}$ query of D as its challenge query. Then the attacker A_2 continues the execution of D by responding to the $(i+1)^{th}$ query using the challenge response, and responds to all remaining queries of D using uniform random responses. Thus, if the challenge response is random then D executes in the $(i+1)^{th}$ hybrid, otherwise it executes in the $(i+2)^{th}$ hybrid. Thus with probability $\mathcal{O}(\epsilon/q^2)$, the attacker A succeeds.

In the other direction, say there is an attacker A with non-negligible success probability in the attack game of definition 18. Then the distinguisher D can simply simulate the attack game of A by responding to all its queries using its function oracle. In response to the challenge query, it either sends the response of its function oracle or a uniform random response. If the function oracle of D is a uniform random function, then A can guess correctly only with probability 1/2, otherwise it has a probability non-negligibly different from 1/2 of guessing correctly. $\square$

In a similar way, we give here an alternative definition for *pseudorandom permutations* (PRP). We will only consider the case of strong PRPs here, hence only give the definition for this case.

Definition 19 ((Strong) Pseudorandom Permutations). *A permutation ensemble* $\{\Pi_\lambda\}_{\lambda\in\mathbb{N}}$ *is a* (strong) pseudorandom permutation *ensemble if for any probabilistic polynomial time (PPT) adversary pair* $A = (A_1, A_2)$, *none of which query their oracles on the challenge query or its inverse, it holds that,*

$$\left| \Pr\left[b = b' \,\middle|\, \begin{array}{l} s \leftarrow \{0,1\}^{c(\lambda)}; (d_{\in\{-1,+1\}}, x, \alpha) \leftarrow A_1^{\Pi_s,\Pi_s^{-1}}(1^\lambda); \\ y_0 \leftarrow \Pi_s^d(x); y_1 \leftarrow \{0,1\}^{b(\lambda)}; b \leftarrow \{0,1\}; \\ b' \leftarrow A_2^{\Pi_s,\Pi_s^{-1}}(y_b, \alpha) \end{array} \right] - \frac{1}{2} \right| = negl(\lambda)$$

This definition is again equivalent to the definition of strong PRPs from chapter 2 (similar to the case of PRFs). A slightly weaker notion than PRFs is that of *Unpredictable functions.* These primitives are similar to *Message Authentication Codes* defined in chapter 2.

Definition 20 (Unpredictable Functions (UF)). *A function ensemble* $\{F_\lambda\}$ *is an* unpredictable function ensemble *if for any probabilistic polynomial time (PPT) adversary A, that does not query its oracle on the prediction query, it holds that,*

$$\Pr\left[y = F_s(x) \;\middle|\; s \leftarrow I(\lambda); (x,y) \leftarrow A^{F_s}(1^\lambda) \right] = negl(\lambda)$$

Similar to the case of PRFs/PRPs, we can also define permutation analogs of UFs, called *Unpredictable permutations.*

Definition 21 (Unpredictable Permutations). *A function ensemble* $\{\Pi_\lambda\}$ *is an* unpredictable permutation ensemble *if for any probabilistic polynomial time (PPT) adversary A, that does not query its oracle on the prediction query or its inverse, it holds that,*

$$\Pr\left[y = \Pi_s(x) \;\middle|\; s \leftarrow \{0,1\}^{c(\lambda)}; (x,y) \leftarrow A^{\Pi_s,\Pi_s^{-1}}(1^\lambda) \right] = negl(\lambda)$$

As we discussed above, we can define verifiable analogs of each of the definitions above. Let us start by defining the notion of *Verifiable Pseudorandom Functions* (VRFs).

Definition 22 (Verifiable Random Functions). *A* Verifiable random function family $\{F_\lambda\}$ *consists of three algorithms* (Gen, Prove, Verify) *such that* $\mathsf{Gen}(1^\lambda)$ *outputs a pair of keys* (PK, SK)*;* $\text{Prove}_{SK}(x)$ *outputs a pair* $(F_{SK}(x), proof_{SK}(x))$*, where* $F_{SK}(x)$ *is the function output and* $proof_{SK}(x))$ *is the corresponding proof of correctness; and* $\text{Verify}_{PK}(x, y, prf)$ *verifies that* $y = F_{SK}(x)$ *using the proof* prf *(by outputting* 1 *if so). This VRF family should satisfy three requirements:*

- **Correctness***: if* $(y, prf) \leftarrow \text{Prove}_{SK}(x)$*, then* $\text{Verify}_{PK}(x, y, prf) = 1$.
- **Soundness***: no* $(PK, x, y_1, prf_1, y_2, prf_2)$*, with* $(y_1, prf_1) \neq (y_2, prf_2)$*, can satisfy*

$$\text{Verify}_{PK}(x, y_1, prf_1) = \text{Verify}_{PK}(x, y_2, prf_2) = 1$$

- **Pseudorandomness***: For any PPT adversary pair* $A = (A_1, A_2)$*, neither of which query their oracle on the challenge input* x*, it holds that*

$$\left| \Pr\left[b = b' \;\middle|\; \begin{array}{c} (PK, SK) \leftarrow \mathsf{Gen}(1^\lambda); \\ (x, \alpha) \leftarrow A_1^{\text{Prove}_{SK}}(1^\lambda); y_0 \leftarrow F_{SK}(x); \\ y_1 \leftarrow \{0,1\}^{b(\lambda)}; b \leftarrow \{0,1\}; \\ b' \leftarrow A_2^{\text{Prove}_{SK}}(y_b, \alpha) \end{array} \right] - \frac{1}{2} \right| = negl(\lambda)$$

Along similar lines, we can define the notions of *Verifiable Pseudorandom Permutations* (VRPs), *Verifiable Unpredictable Functions* (VUFs) and *Veri-*

fiable Unpredictable Permutations (VUPs) as verifiable analogs of PRPs, UFs and UPs, respectively, each of which has three algorithms (Gen, Prove, Verify), and satisfy the Completeness and Soundness properties as well.

6.2 Implications

In the previous chapter, we proved a combinatorial property of the Feistel construction where internal round function values were visible to the adversary. Now we will describe how this property can be applied to a variety of scenarios to yield new or improved cryptographic constructions than before.

6.2.1 More Resilient PRPs from PRFs

We give a construction of *pseudorandom permutations* from *pseudorandom functions*, that remains secure even if the PRF input/output pairs used in the construction are visible to the attacker. In particular, we propose using a k-round LR construction $\Psi_{R,k}$, where $k = \omega(\log \lambda)$, with independent PRFs $f_1 \ldots f_k \leftarrow F$ as round functions. The following states that this construction is a secure PRP even if the attacker can observe the intermediate round values.

Theorem 25. *If there exists an efficient PRP adversary A_π that has a non-negligible advantage ϵ_π in the PRP attack game against the construction $\Psi_{R,k}$ (using round functions from the PRF family F), then there also exists a PRF adversary A_f that has non-negligible advantage ϵ_f in the PRF attack game* *against a PRF sampled from the PRF family F. From this, we get a bound $\epsilon_\pi = \mathcal{O}\left(qk\epsilon_f + \frac{(qk)^6}{2^n}\right)$, where ϵ_f denotes the maximum advantage of a PRF adversary running in time $\mathcal{O}(t + (qk)^5)$ against a PRF sampled from F, and t, q are the running time and number of queries made by A_π.*

Proof: We show that the PRP construction $\Psi_{R,k}$, using PRFs sampled from the PRF family $F_{(\cdot)} : \{0,1\}^n \to \{0,1\}^n$, is a secure PRP. The proof consists of two parts:

1. Showing that a PRF family that yields secure and independent PRFs upon each sample is a 5-XOR resistant function family.

2. Showing that no PRP adversary can succeed with non-negligible advantage in the PRP attack game against a $\omega(\log \lambda)$-round Feistel construction with independent and secure PRFs in each round.

XOR-resistance of PRFs. Consider a k-round Feistel construction Ψ_k that uses k PRFs $f_1 \dots f_k$, independently sampled from a PRF family $F_{(\cdot)} : \{0,1\}^n \to \{0,1\}^n$, as round functions. Consider an XOR finding adversary A_{xor} that forces a 5-XOR condition through its queries with non-negligible advantage. We will show that using A_{xor}, we can design another attacker A_f that succeeds in the PRF attack game (see definition 18) against a PRF sampled from the family F.

Claim 26. *If there is an PPT attacker A_{xor} that queries the k-round Feistel construction Ψ_k (that uses independent PRFs from a PRF family F), observes intermediate round values and forces the 5-XOR condition through its* *queries with probability ϵ_{xor}, then there exists a PRF adversary A_f that has advantage ϵ_f in the PRF attack game against a PRF sampled from F, where $\epsilon_f \geq \frac{1}{2qk} \cdot \left(\epsilon_{xor} - \frac{(qk)^6}{2^n}\right)$.*

proof of claim 26: The PRF adversary A_f gets oracle access to the challenge PRF adversary F_s. It then needs to choose a challenge query, to which it either gets the PRF output or a random n-bit string, and its task is to distinguish between the two cases. The attacker A_f chooses a random round number $i \in \{1 \dots k\}$, and samples $(k-1)$ independent PRFs $f_1 \dots f_{i-1}, f_{i+1} \dots f_k$ from the family F. It then simulates the Feistel construction Ψ_k, with the challenge PRF as the i^{th} round function and the self-generated PRFs making up the remaining round functions. It then simulates the XOR attack game between A_{xor} and Ψ_k.

Assume a fixed, large enough, polynomial upper bound on the number of queries that the adversary A_{xor} makes to Ψ_k. The PRF adversary chooses a random query number $j \in \{1 \dots q\}$ where it chooses its challenge query. On getting the j^{th} query, it sends the i^{th} round value as the challenge PRF query, and uses the challenge response as the output of the i^{th} round function. It computes the remaining self-generated round functions as usual. If the i^{th} round function is applied to a new input, then it checks to see if the new round value generated has a 5-XOR representation in terms of previously existing round values. If so, then it guesses that the challenge response is the PRF output (say by outputting 1), otherwise it guesses the challenge response to be random (by outputting 0).

It is clear that if the attacker A_f makes all its guesses correctly, i.e. correct round number and correct query number, then it succeeds if a random response also does not have an 5-XOR representation. Hence, we get that

$$\begin{aligned} Adv(A_f) &= \Pr[(A_f \to 1) \wedge (\text{PRF output})] \\ &\quad + \Pr[(A_f \to 0) \wedge (\text{Random output})] - \frac{1}{2} \end{aligned}$$

Here $A_f \to 0/1$ represents the event that the attacker A_f outputs 0/1. If ϵ_{xor} denotes the advantage of an XOR adversary then we get that,

$$\begin{aligned} \Pr[(A_f \to 1) \wedge (\text{PRF output})] &= \Pr[(A_f \to 1)|(\text{PRF output})] \\ &\quad \cdot \Pr[(\text{PRF output})] \\ &\geq \frac{\epsilon_{xor}}{qk} \cdot \frac{1}{2} \\ \Pr[(A_f \to 0) \wedge (\text{Random output})] &= \Pr[(A_f \to 0)|(\text{Random output})] \\ &\quad \cdot \Pr[(\text{Random output})] \\ &\geq \left[1 - \frac{(qk)^5}{2^n}\right] \cdot \frac{1}{2} \\ \Rightarrow Adv(A_f) &\geq \frac{1}{2qk} \cdot \left[\epsilon_{xor} - \frac{(qk)^6}{2^n}\right] \end{aligned}$$

□

Security of the PRP construction. We will now show that the construction $\Psi_{R,k}$, that is based on a k-round Feistel construction using independently sampled round functions from a PRF family $F_{(\cdot)} : \{0,1\}^n \rightarrow \{0,1\}^n$, is a secure PRP construction.

Claim 27. *If there exists an efficient PRP adversary A_π that has a non-negligible advantage ϵ_π in the PRP attack game against the construction $\Psi_{R,k}$ (using round function from PRF family F), then there also exists a PRF adversary A_f that has non-negligible advantage ϵ_f in the PRF attack game against a PRF sampled from the PRF family F. In particular, we get that the maximum advantage of such a PRP adversary can be bounded by $\epsilon_\pi = \mathcal{O}\left(qk\epsilon_f + \frac{(qk)^6}{2^n}\right)$. Here ϵ_f is the maximum advantage of a PRF adversary running in time $\mathcal{O}(t + (qk)^5)$ against a PRF sampled from F, where t, q are the running time and number of queries made by A_π.*

proof of claim 27: The PRF adversary A_f gets oracle access to a challenge PRF F_s. It samples $(k-1)$ independent PRFs $f_1 \dots f_{(k/2)-1}, f_{(k/2)+1} \dots f_k$ from the PRF family F. It simulates the PRP construction $\Psi_{R,k}$ by plugging in the challenge PRF as the $(k/2)^{th}$ round function and using the self-generated PRFs as the remaining round functions. It then simulates the PRP attack game between the attacker A_π and $\Psi_{R,k}$.

It computes the response to any query made by A_π by computing all the round values ($(k/2)^{th}$ one by querying the PRF oracle). When the attacker A_π sends its challenge query, then A_f computes all the self-generated round functions honestly, but sends the $(k/2)^{th}$ round value as its PRF challenge query and uses the challenge query response as the $(k/2)^{th}$ round function output. It then continues with the post-challenge phase as it did before the challenge query. Finally, it simply gives the same output as the PRP adversary A_π (i.e. guess PRF if A_π guesses PRP, else guess random).

We note that the PRF adversary succeeds if the following conditions all hold: (1) the PRP adversary A_π succeeds, (2) the $(k/2)^{th}$ round value in the challenge query is never required in any other query and, (3) the PRP challenge output looks random if the PRF challenge response is random. Thus, we have that,

$$Adv(A_f) \geq \Pr\left[\begin{array}{l}(A_\pi \text{ succeeds})\\ \wedge(\text{no } (k/2)^{th} \text{ round collision})\\ \wedge(\text{PRF random}\Rightarrow \text{ PRP random})\end{array}\right] - \frac{1}{2}$$

Now we can estimate the probability in the above expression as,

$$\begin{array}{rl}
& \Pr\left[\begin{array}{l}(A_\pi \text{ succeeds}) \wedge (\text{no } (k/2)^{th} \text{ round collision})\\ \wedge(\text{PRF random}\Rightarrow \text{ PRP random})\end{array}\right]\\
\geq & \Pr\left[\begin{array}{l}(A_\pi \text{ succeeds})\\ \wedge(\text{PRF random}\Rightarrow \text{ PRP random})\end{array}\middle| \left(\begin{array}{c}\text{no collision in}\\ \{\frac{k}{2}-1, \frac{k}{2}+1\}\end{array}\right)\right]\\
& \cdot \Pr[\text{no collision in } \{\frac{k}{2}-1, \frac{k}{2}+1\}]\\
\geq & \Pr\left[(A_\pi \text{ succeeds}) \middle| \begin{array}{l}(\text{PRF random}\Rightarrow \text{ PRP random})\\ \wedge((\text{no collision in } \{\frac{k}{2}-1, \frac{k}{2}+1\})\end{array}\right]\\
& \cdot \Pr\left[(\text{PRF random}\Rightarrow \text{ PRP random}) \middle| \left(\begin{array}{c}\text{no collision in}\\ \{\frac{k}{2}-1, \frac{k}{2}+1\}\end{array}\right)\right]
\end{array}$$

$$
\begin{array}{rcl}
 & & \cdot \Pr[\text{no collision in } \{\frac{k}{2}-1, \frac{k}{2}+1\}] \\
 & \geq & \Pr\left[(A_\pi \text{ succeeds}) \middle| \begin{array}{l} (\text{PRF random} \Rightarrow \text{ PRP random}) \\ \wedge((\text{no collision in } \{\frac{k}{2}-1, \frac{k}{2}+1\}) \end{array} \right] \\
 & & \cdot \Pr\left[(\text{PRF random} \Rightarrow \text{ PRP random}) \middle| \left(\begin{array}{c} \text{no collision in} \\ \{\frac{k}{2}-1, \frac{k}{2}+1\} \end{array} \right) \right] \\
 & & \cdot(1-\epsilon_{xor}) \\
 & \geq & \Pr\left[(A_\pi \text{ succeeds}) \middle| \begin{array}{l} (\text{PRF random} \Rightarrow \text{ PRP random}) \\ \wedge((\text{no collision in } \{\frac{k}{2}-1, \frac{k}{2}+1\}) \end{array} \right] \\
 & & \cdot(1-2\epsilon_f)\cdot(1-\epsilon_{xor}) \\
 & \geq & \left(\epsilon_\pi - 2\epsilon_f \cdot \left(2qk\epsilon_f + \frac{(qk)^6}{2^n}\right)\right)\cdot(1-2\epsilon_f)\cdot\left(1-2qk\epsilon_f - \frac{(qk)^6}{2^n}\right) \\
\Rightarrow \epsilon_\pi & = & \mathcal{O}\left(qk\epsilon_f + \frac{(qk)^6}{2^n}\right)
\end{array}
$$

In the above argument, we have used ϵ_f to bound the advantage of all of out PRF adversaries (in L.H.S. as well as R.H.S.). This bound ϵ_f is the maximum advantage of a PRF adversary running in time $\mathcal{O}(t+(qk)^5)$, where t, q are the running time and number of queries made by the PRP attacker A_π. The initial two steps of the above argument can be derived as simple conditional probability manipulations. The third step can be derived as a result of corollary 2 in chapter 5, that says that the advantage of the collision finding attacker is no more than that of a 5-XOR finding attacker.

In the fourth step, we use the fact that if a PRF family yields secure and independent PRFs, then the usual PRF attack definition is equivalent to a modified definition where the attacker has access to two independently sampled PRFs from the same family. In the challenge phase of this new attack scenario, either random or pseudorandom responses are given to challenge queries to both these functions. Since the attacker is not permitted to query the PRF oracles on these challenge queries, we need the property that no round value collision occur among round values in $\{(k/2)-1\ldots(k/2)+1\}$. □

⊒

Moreover, since we know from theorem 18 in chapter 5 that there is an attacker that can invert the Feistel network with $k = \mathcal{O}(\log\lambda)$ rounds within a polynomial number of forward queries, we can also deduce that the result above is asymptotically tight.

6.2.2 Unpredictable Permutations

What if the round functions in the Feistel network are only *unpredictable functions* and not pseudorandom? In this case, it is not clear whether the attack in theorem 18 (chapter 5) can be made to work in this case. This is because the UP adversary cannot make use of this attacker since it does not seem to have access to all the intermediate round values. However, we will first show that if certain pathological (but secure) unpredictable functions are used as round functions, then the UP adversary can infer *all* the round values simply by observing the output of the Feistel construction!

Lemma 23. *For any $k \leq \frac{n}{\omega(\log\lambda)}$ (in particular, if $k = \mathcal{O}(\log\lambda)$), there exist*

k secure unpredictable functions $f_1 \dots f_k$, such that by querying the k-round Feistel construction $\Psi_{f_1 \dots f_k}$ on any input an efficient attacker can always learn all intermediate round values.

Proof: Let $\{g_i : \{0,1\}^n \to \{0,1\}^{n/k}\}_{i \in \{1 \dots k\}}$ be k secure unpredictable functions. For $i \in \{1, k\}$, we will define the functions $f_i : \{0,1\}^n \to \{0,1\}^n$ as $f_i(x) = 0^{(i-2)\cdot(n/k)} \parallel x_{i-1} \parallel g_i(x) \parallel 0^{(k-i)\cdot(n/k)}$, where x_{i-1} denotes the $(i-1)^{th}$ (n/k) bit block in the input x. Each of the functions f_i is a secure unpredictable function if the corresponding function g_i is a secure UF.

Consider a query $(R_0 \parallel R_1) \in \{0,1\}^{2n}$ made to the Feistel construction $\Psi_{f_1 \dots f_k}$. We will consider k blocks of (n/k) bits each in both R_0 and R_1, which we will denote by $R_0 = R_0^1 \parallel \dots \parallel R_0^k$ and $R_1 = R_1^1 \parallel \dots \parallel R_1^k$. Denote the round values generated in computing the output of this construction as $(R_0, R_1) \dots (R_k, R_{k+1})$, where $R_k \parallel R_{k+1}$ is the output of this construction. If the number of rounds in the Feistel construction is even, then we note that the output of the construction is:

$$\begin{aligned}
R_k &= (g_1(R_1) \oplus R_0^1 \oplus R_1^1) \parallel \dots \parallel (g_{k-2}(R_{k-2}) \oplus R_0^{k-2} \oplus R_1^{k-2}) \\
&\quad \parallel (g_{k-1}(R_{k-1}) \oplus R_0^{k-1}) \parallel R_0^k \\
R_{k+1} &= (g_1(R_1) \oplus R_0^1 \oplus R_1^1) \parallel \dots \parallel (g_{k-1}(R_{k-1}) \oplus R_0^{k-1} \oplus R_1^{k-1}) \\
&\quad \parallel (g_k(R_k) \oplus R_1^k)
\end{aligned}$$

If number of rounds k is odd, then the output of the Feistel construction is,

$$\begin{aligned} R_k &= (g_1(R_1) \oplus R_0^1 \oplus R_1^1) \parallel \ldots \parallel (g_{k-2}(R_{k-2}) \oplus R_0^{k-2} \oplus R_1^{k-2}) \\ &\quad \parallel (g_{k-1}(R_{k-1}) \oplus R_1^{k-1}) \parallel R_1^k \\ R_{k+1} &= (g_1(R_1) \oplus R_0^1 \oplus R_1^1) \parallel \ldots \parallel (g_{k-1}(R_{k-1}) \oplus R_0^{k-1} \oplus R_1^{k-1}) \\ &\quad \parallel (g_k(R_k) \oplus R_0^k) \end{aligned}$$

Now it is easy to find each of the round function outputs (and hence the intermediate round values) by simply observing the right half of the output of the Feistel construction. $\square$

Thus we see that if the number of rounds in the Feistel construction (using UFs) used to construct *unpredictable permutations* is $k = \mathcal{O}(\log \lambda)$, then the construction is insecure. Moreover, even if we attempt to construct a shrinking MAC by chopping the left half of the output, it would be possible to retrieve all intermediate round values by simply observing the MAC output. In fact, even for $k = \omega(\log \lambda)$ (but less than $n/\omega(\log \lambda)$) rounds it might be possible to retrieve all intermediate round values, and hence none of the previous proof techniques are applicable. We will prove a much stronger result here, by showing that if we use a super-logarithmic number of rounds in the Feistel construction (with independent unpredictable round functions) then the resulting construction is an unpredictable permutation even if the adversary gets all the intermediate round values along with the permutation output (which, as we saw, it may get any way for some pathological UFs).

The UP construction $\Psi_{U,k}$ consists of $k = \omega(\log \lambda)$ rounds of the Feistel transformation using independent UFs $f_1 \dots f_k \leftarrow F$. The following theorem essentially states that this construction is a secure UP construction.

Theorem 28. *If there exists an efficient UP adversary A_π that has non-negligible advantage ϵ_π in the unpredictability game against $\Psi_{U,k}$ and which makes a polynomial number of queries to $\Psi_{U,k}$, then there also exists a UF adversary A_f that has non-negligible advantage in the unpredictability game against a UF sampled from the UF family F. From this, we get that the maximum advantage of the UP adversary A_π is $\epsilon_\pi = \mathcal{O}\left(\epsilon_f \cdot (qk)^6\right)$. Here ϵ_f denotes the maximum advantage of a UF adversary running in time $\mathcal{O}(t + (qk)^5)$ against a UF sampled from F, where t is the running time of the PRP adversary A_π and q is the number of queries made by it.*

Proof: The proof of this theorem consists of two main parts:

1. A UF family that yields secure and independent UFs on each sample is a 5-XOR resistant function family.

2. The construction $\Psi_{U,k}$ that uses secure and independent UFs in each round is a secure *unpredictable permutation.*

XOR-resistance of UFs. Consider the k-round Feistel construction $\Psi_{U,k}$ using independent UFs $f_1 \dots f_k \leftarrow F$ in each round. If there is an adversary A_{xor} that queries $\Psi_{U,k}$ and forces a 5-XOR condition through its queries with a non-negligible advantage ϵ_{xor}, then we can construction a UF adversary A_f

that has non-negligible advantage in the unpredictability game against a UF sampled from the family F.

Claim 29. *If there is an adversary A_{xor} that can force a 5-XOR condition in an interaction with $\Psi_{U,k}$ (that uses independent UFs sampled from a UF family $F_{(\cdot)} : \{0,1\}^n \rightarrow \{0,1\}^n$) with non-negligible probability ϵ_{xor} then there exists a VUF adversary A_f that has non-negligible success probability ϵ_f in the unpredictability against a UF sampled from the family F. In particular, we show that $\epsilon_f \geq \frac{\epsilon_{xor}}{(qk)^6}$.*

proof of claim 29: On getting the challenge unpredictable function F_s, the UF adversary chooses a random round number i where it plugs in the challenge UF. Next, the UF adversary A_f generates $(k-1)$ independent UFs $f_1 \dots f_{i-1}, f_{i+1} \dots f_k$ from the same family and uses these as the remaining round functions to simulate the Feistel construction $\Psi_{U,k}$ for the XOR adversary A_{xor} to attack.

Then it lets the UF adversary run its attack on $\Psi_{U,k}$. Assuming a fixed and large enough polynomial upper bound q on the number of queries made by A_{xor}, the UF adversary A_f chooses a random query number $j \in \{1, q\}$. It guesses that the 5-XOR condition occurs after the i^{th} round function evaluation in the j^{th} query, i.e. R_i^j. Instead of querying this input to the UF oracle, it selects this as the challenge input and uses an XOR of upto 5 randomly chosen previously existing round values as its prediction of the output.

If all its guesses are correct, i.e. it chooses the correct round number i, the correct query number j and the correct XOR representation, then it succeeds in the UF game. The probability that all its guesses are correct is at least $\frac{1}{(qk)^6}$. Thus, we get that $\epsilon_f \geq \frac{\epsilon_{xor}}{(qk)^6}$. □

Security of the UP construction. We will now show that the UP construction $\Psi_{U,k}$, that uses round functions from the UF family F that gives secure and independent UFs on each sample, is a secure construction of a *unpredictable permutation*

Claim 30. *If there exists a PPT UP adversary A_π that has non-negligible advantage ϵ_π in the unpredictability game against $\Psi_{U,k}$ and which makes a polynomial number of queries to $\Psi_{U,k}$, then there also exists a UF adversary A_f that has non-negligible advantage in the unpredictability game against a UF sampled from the UF family F. In particular, we get that the maximum advantage of the UP adversary A_π is $\epsilon_\pi = \mathcal{O}(\epsilon_f \cdot (qk)^6)$. Here ϵ_f is the maximum advantage of a UF adversary running in time $\mathcal{O}(t + (qk)^5)$ against a UF sampled from F, where t, q are the running time and number of queries made by A_π.*

proof of claim 30: The UF adversary A_f gets oracle access to a challenge unpredictable function F_s. It samples $(k-1)$ independent UFs $f_1 \ldots f_{(k/2)-1}$, $f_{(k/2)+1} \ldots f_k$ from the same UF family F. It simulates the UP construction by plugging in the challenge UF as the $(k/2)^{th}$ round function, and using the self-generated UFs as the other round functions.

When the UP adversary sends its challenge query $R_0 \parallel R_1$ (or $R_k \parallel R_{k+1}$), and its predicted output $R_k \parallel R_{k+1}$ (resp. $R_0 \parallel R_1$), the UF adversary proceeds by using its self generated round functions to evaluate the intermediate round values $R_0, R_1, R_2 \dots R_{k/2}$ and from $R_{k+1}, R_k, R_{k-1} \dots R_{k/2+1}$. It then sends the challenge input/output pair $(R_{k/2}, R_{k/2-1} \oplus R_{k/2+1})$ as its prediction. It is easy to see that if the round value $R_{k/2}$ is a new round value and the UP adversary predicted correctly, then the UF adversary A_f succeeds. Thus, we can deduce that,

$$\begin{aligned}
\Pr[A_f \text{ succeeds}] &= \Pr\left[\begin{array}{l}(A_\pi \text{ succeeds}) \\ \wedge(\text{no } (k/2)^{th} \text{ round collision})\end{array}\right] \\
&= \Pr\left[(A_\pi \text{ succeeds}) \,\middle|\, \text{no } (k/2)^{th} \text{ round collision}\right] \\
&\quad \cdot \Pr\left[\text{no collision}\right] \\
&\geq \left(\Pr\left[A_\pi \text{ succeeds}\right] - \Pr\left[(k/2)^{th} \text{ collision }\right]\right) \\
&\quad \cdot \Pr\left[\text{no collision}\right] \\
&\geq (\epsilon_\pi - \epsilon_{xor}) \cdot (1 - \epsilon_{xor}) \\
\Rightarrow \epsilon_\pi &\leq \frac{\epsilon_f}{1 - \epsilon_{xor}} + \epsilon_{xor} \\
\Rightarrow \epsilon_\pi &= \mathcal{O}(\epsilon_f \cdot (qk)^6)
\end{aligned}$$

In the above argument, we have often bound the advantage of a UF adversary by ϵ_f. This is the maximum advantage of a UF adversary running in time $\mathcal{O}(t + (qk)^5)$, where t, q are the running time and number of queries made by A_π. The transition from step (3) to (4) is possible using corollary 2 from chapter 5, that says that the advantage of an efficient collision finding adversary is same as that of an efficient XOR condition forcing adversary. The last step of the argument is possible through claim 29. □

⊒

6.2.3 Verifiable Random Permutations

When we attempt to use the Feistel network to construct a *verifiable random permutation* using VRFs as round functions, then the attacker gets all the intermediate round values as part of the proofs for each round function computation. Thus, here again, one can use the attacker from lemma 18 (chapter 5) to construct a VRP attacker that violates the pseudorandomness requirement of the VRP construction if the number of Feistel rounds is $k = \mathcal{O}(\log \lambda)$.

The VRP construction$\Psi_{VR,k}$ that we use is the k-round LR construction using independent VRFs $f_1 \dots f_k \leftarrow F$ as round functions. The public/private keys of $\Psi_{VR,k}$ are simply the concatenation of the public/private keys of the k VUFs. The Prove functionality for $\Psi_{VR,k}$ gives the permutation output, and as proof it gives all intermediate round values along with the corresponding VRF proofs. The Verify functionality simply checks if all intermediate VRF proofs verify correctly.

Theorem 31. *Let $\Psi_{VR,k} = (G_\pi, \Pi, V_\pi)$ be the VRP construction that uses a k-round LR construction with independent VRFs $f_1 \dots f_k \leftarrow F$. This construction is a secure VRP if the VRFs used as round functions are secure. In particular, for any probabilistic polynomial time oracle machine $A_\pi = (A_1, A_2)$ that does not query its oracle on x or try to invert the response to the challenge query, the advantage of A_π in winning the VRP pseudorandomness game against $\Psi_{VR,k}$ is at most $\mathcal{O}\left(qk\epsilon_f + \frac{(qk)^6}{2^n}\right)$, where ϵ_f denotes the maximum advantage of a VRF adversary that runs in time $\mathcal{O}(t + (qk)^5)$ against a VRF sampled from F, and t and q are the running time and number of queries made by A_π.*

Proof: The completeness of the construction $\Psi_{VR,k}$ is a direct consequence of the completeness of each of the VRFs used as round functions, since if all VRF proofs verify correctly then the resulting VRP proof does so too. The *soundness* of the construction if also obvious given the fact that all the intermediate VRFs are sound. In particular, if there are two output/proof pairs of $\Psi_{VR,k}$ that verify correctly, then we can find two VRF output/proof pairs that verify correctly for one of the round functions. The proof for the *pseudorandomness* property of $\Psi_{VR,k}$ is essentially the same as the proof of theorem 25. $\square$

6.2.4 Verifiable Unpredictable Permutations

Our VUP construction $\Psi_{VU,k}$ is essentially identical to the VRP construction $\Psi_{VR,k}$, excepts that we use independent VRFs instead of VUFs as round functions.

Theorem 32. *Let $\Psi_{VU,k} = (G_\pi, \Pi, V_\pi)$ be the construction using k rounds of the Feistel construction using independent VUFs $f_1 \dots f_k \leftarrow F$. Then $\Psi_{VU,k}$ is a secure VRP if the VUFs used in the construction are secure VUFs. In particular, for any probabilistic polynomial time oracle machine A_π that does not make a forward query on x or an inverse query on y, the advantage of A_π in winning the VUP pseudorandomness game against $\Psi_{VU,k}$ is at most $\mathcal{O}(q^6 k^7 \cdot \epsilon_f)$, where ϵ_f denotes the maximum advantage of a VUF adversary running in time $\mathcal{O}(t+(qk)^5)$ against a VUF sampled from F, t is the running time of A_π and q is the number of queries made by A_π.*

The completeness and soundness properties of this construction can be proven in the same way as the corresponding properties of the VRP construction $\Psi_{VR,k}$, above. The proof for the *unpredictability* property is the same as in theorem 28.

6.3 Applications

We have seen that our results for the Feistel network with public round values leads to new or improved constructions of several cryptographic primitives. In this section, we will illustrate several practically-motivated natural scenarios where our results are applicable.

6.3.1 Implications to Domain Extension

Since the Feistel Network doubles the length of its input, our results could also be viewed in relation to the question of domain extension of UFs, VUFs and VRFs. In practice, the question of domain extension is typically handled by a collision-resistant hash function (CRHF): it uses only one call the the underlying n-bit primitive f and does not require the secret key to grow. However, the existence of a CRHF is a theoretically strong assumption, which does not seem to follow from the mere existence of UFs, VRFs or VUFs. This is especially true for UFs, whose existence follows from the existence of mere one-way functions and, hence, can even be "black-box separated" from CRHFs [71]. Thus, it makes sense to consider the question of domain extension *without introducing new assumptions.*

For PRFs, this question is easily solved by using (almost) universal hash functions (instead of CRHFs) to hash the message to n bits before applying the n-bit PRF. However, this technique fails for UFs, VUFs and VRFs: in the case of unpredictability because the output reveals information about the hash key, and for VRFs because it is unclear how to provide proofs of correctness without revealing the hash key. Another attempt (which works for digital signatures) is to use target collision-resistant hash functions [60] in place of CRHFs, but such functions have to be freshly chosen for each new input, which will break the unique provability of UFs, VUFs and VRFs. (Additionally, the hash key should also be authenticated, which further decreases the bandwidth). In case the underlying n-bit primitive f is shrinking (say, to $n - a$ bits), one can use some variant of the cascade (or Merkle-Damgård) construction. Indeed, this was formally analyzed for MACs by [1, 53]. However, the cost of this method is one evaluation of f per a input bits. In particular, in case the output of f is also equal to n, which is natural if one wants to extend the domain of a UF given by a block cipher, this method is either inapplicable or very inefficient.[1]

In contrast, our method builds a UP/VUP/VRP from $2n$ to $2n$ bits from the one from n to n bits, by using $k = \omega(\log \lambda)$ evaluations of f, albeit also at the price of increasing the secret key by the same amount. This answers the question left open by An and Bellare [1] (who only showed that three rounds are insufficient): *Feistel Network is a good domain extender for MACs if and only if it uses super-logarithmic number of rounds*!

Moreover, in the context of UFs (and VUFs), where one wants to minimize the output length as well, we notice that the output length can be easily reduced from $2n$ to n. This is done by simply dropping the "left half" of the Feistel permutation output! The justification for this optimization follows by noticing that in this case the attacker will only make forward queries to the Feistel construction. For such attackers, we can extend our main combinato-

[1]In principle, such length-preserving f can be "truncated" by a bits, but this loses an exponential factor in a in terms of exact security. Thus, to double the input length, one would have to evaluate f at least $\Omega(n/\log \lambda)$ times.

rial lemma as follows. For any $s \leq k$, if a 5-XOR resistant family is used to implement the round functions and the attacker made less than exponential in s number of queries, then the attacker has a negligible chance to cause any collisions between rounds s and k (as opposed to $k - s$ we had when backward queries were allowed). From this, one can derive that $k = \omega(\log \lambda)$ Feistel rounds is enough to turn a UF (or VUF) from n to n bits into one from $2n$ to n bits. Moreover, in the case of UFs we expect that one would use a (possibly heuristic) pseudorandom generator to derive the k round keys (much like in the case of block ciphers), meaning that the only effective cost is k computations of the basic UF. Once the domain is doubled, however, one can use the cascade methods [1, 53] to increase it further without increasing the key or the output length.

6.3.2 More Resilient Block Ciphers

Although not as strong as pseudorandomness, unpredictability is a meaningful property of block ciphers. First, we already mentioned that it is enough for message authentication, and our Feistel construction is also useful in the context of domain extension of MACs. We notice that it is also enough to argue certain weaker properties of popular modes of operation on block ciphers. For example, one can easily argue that the CBC mode with UPs (rather than PRPs) yield a "computationally ϵ-universal" hash function [3], which can then be used with an ordinary block cipher to get a secure "encrypted CBC-MAC". Even in the context of encryption, one can argue that CBC, OFB and CFB modes with UPs satisfy the following form of one-wayness against the usual chosen message attack. The attacker can ask encryptions or any messages. For the challenge, it specifies any message with one missing block. Then this block is chosen at random, and the encryption of the entire message (using the corresponding mode) is given to the attacker. Finally, the attacker has to recover this missing block, and using UPs guarantees that the attacker only has a negligible probability to succeeded in this game.

To summarize, the usage of UPs in place of PRPs still maintains weaker, but still meaningful security properties. Therefore, we see their primary utility as a way for providing a "graceful fall-back" property for the Feistel construction. If (nearly) pseudorandom round functions are used, then with $\omega(\log \lambda)$ rounds the resulting permutation is a PRP. As a bonus, it remains a PRP even if the intermediate round values could be leaked! Additionally, even if the round functions are only unpredictable, we still have some basic security left, so at the very least the system will not be "completely broken".

6.3.3 Ideal Cipher Model using Semi-Honest Trusted Party

The *Ideal Cipher Model* (ICM) (also known as the "Shannon Model") assumes the existence of a publicly accessible Ideal Block Cipher, meaning that for every possible key s one has a fresh random permutation Π_s and its inverse Π_s^{-1}. Although the ICM is not as popular as the random oracle model, there are still several notable examples of schemes where this model has been used [15, 23, 30, 42, 46]. Unfortunately, just like the random oracle model, the ICM model cannot be provably realized without a trusted party T (see [14]). A naive implementation is easy, but inconvenient. First, T should keep track of all the queries already asked to ensure consistency, which quickly becomes very impractical. Second, the parties must trust that T has evaluated the value $\Pi_s(x)$ consistently across invocations. Third, once they get such a value, they cannot convince any other party of its validity: that party must independently go to T to check the correctness. Finally, they must trust that the answers of T are actually random.

It turns out that a VRP can considerably improve this naive implementation. First, we start with implementing a single truly random permutation Π (corresponding to an ideal cipher with a fixed key). Then T can publish the public key for a VRP π, and only keep the secret key as its state. When some party comes to T and asks a forward or backward query to Π, T simply evaluates π or π^{-1} on that query, and returns the result together with a proof of correctness. This way the parties are assured that: (a) they receive a correct and *consistent* value of Π; (b) they are really talking to T (or, if not, the value is correct anyway); (c) once T is committed to the public key, T cannot dynamically adjust the values of Π and Π^{-1}; (d) even if T selected a bad public key, T is committed to a *permutation*; in particular, the value of Π on a *random* point is guaranteed to be random. Finally, once somebody gets a value of Π or Π^{-1} from T, it can transfer this value on its own, without the need of other parties to come to T and verify it.

To extend this to a full blown Ideal Cipher, we face a problem that T must generate a new VRP for every key s of the Ideal Cipher. However, for our particular VRF-based construction we can do better. Instead of assuming the existence of a VRF from n to n bits, we assume the existence of a VRF from $n + a$ to n bits, where a is the length of the key s (if needed, such VRF can always be constructed from another VRF using the domain extension techniques we developed earlier). In this case, T will always prepend the key s to all the VRFs inputs when evaluating the Feistel Network for the value of Π_s. This way T still stores only $\omega(\log \lambda)$ keys for the VRFs, and can emulate 2^a possible random ciphers.

6.3.4 Applications of VRPs/VUPs

Next, we mention several examples how VRPs could be useful in scenarios where plain VRFs are not enough.

Non-interactive Commitments

We notice that VRPs immediately yield non-interactive, setup-free, perfectly-binding commitments schemes. The sender chooses a random key pair (SK, PK) for a VRP π. To commit to m (in the domain of the VRP), the sender sends PK and the value $c = \pi_{SK}(m)$ to the receiver. To open m, the sender sends m and the proof that $c = \pi_{SK}(m)$, which the receiver can check using the public key PK. The hiding property of this construction trivially follows for the security of VRPs. As for binding, it follows from the fact that π is a permutation even for an *adversarial choice of PK*. As we can see, it is not clear how to achieve binding *directly* using plain VRFs. However, given our (non-trivial) equivalence between VRFs and VRPs, we get that VRFs are also sufficient for building non-interactive, perfectly binding commitment schemes without setup. Alternatively, to commit to a single bit b, one can use VUPs augmented with the Goldreich-Levin bit [37]. Here the sender would pick a random r and x, and send PK, r, $\pi_{SK}(x)$, and $(x \cdot r) \oplus b$, where $x \cdot r$ denotes the inner product modulo 2. Using our equivalence between VUPs and VUFs, we see that VUFs are sufficient as well.

We remark that the best general constructions of such commitments schemes was previously based on one-way permutations (using the hardcore bit) [16], since Naor's construction from one-way functions [57] is either interactive, or non-setup-free. Since the assumption of one-way permutations is incompatible with VUFs or VRFs, our new construction is not implied by prior work.

Non-Interactive Zero-Knowledge (NIZK)

We show that VRPs (and, thus, indirectly, VRFs), could be used to construct NIZK proofs (in the common reference string model). We remark, however, that Dwork and Naor [29] already gave a completely different construction of NIZK proofs from VRFs (and even a weaker primitive called *verifiable pseudorandom generator*). Thus, our construction only gives an alternative (and different) proof of an already known result by [29]. Nonetheless, we believe that it naturally illustrates the usefulness of VRPs in comparison to VRFs, and also solves a question left open by Goldwasser and Ostrovsky [38] (see below).

Feige et al. [31] reduced the question of constructing NIZK proofs (in the common reference string model) to the question of implementing the so called "hidden bits system" HBS, and showed how to implement HBS using trapdoor permutations. Later, Goldwasser and Ostrovsky [38] showed how to implement HBS using so called *invariant signatures*. In our modern terminology, invariant signatures are quite similar to VRFs, except for one additional requirement: they should induce a (pseudo)random distribution on the output when applied to a random input, *even if the public key for the VRF is adversarially chosen*. Thus, we can think of *invariant signatures* as "balanced" VRFs. Unfortunately, it is easy to see that regular VRFs are not enough to plug into the construction of [38]. Namely,

(a) Plain VRFs do not have to satisfy this property (and, as far as we can see, there is no trivial way to enforce it in VRFs; although, our results imply a non-trivial way to do so).

(b) More severely, there exist secure (and, of course, unbalanced) VRFs for which the transformation of [38] is completely insecure.

To briefly see point (a), imagine adding a new special public key PK^* to any secure VRF, for which the VRF is defined to be identically zero. It is clear that this still defines a VRF, since the prover is still committed to a unique function, even for the key PK^*. And pseudorandomness holds, since the chances PK^* will be selected are negligible. Yet, the new VRF is obviously unbalanced. In fact, if we use this new VRF in place of the invariant signature in the construction of [38], we will get a completely insecure HBS system (thus, showing (b)). Briefly, in the construction of [38] a VRF *selected by the prover* is applied to a bunch or random points to define the "hidden random string" (for which the prover can selectively open some part later). If the prover chooses PK^* as his public key, then the hidden random string is all zero as well, and it is easy to see that NIZK construction of [31] will completely fails with such non-random HRS.

On a positive side, VRPs trivially satisfy balancedness, since they are guaranteed to be permutations for any value of the public key. This means one can build NIZK proofs from VRPs. By our construction of VRPs from VRFs, we see that VRFs are also sufficient for NIZK proofs for NP. Also, even VUPs coupled with the Goldreich-Levin bit turn out to be sufficient for this application.

Non-interactive Lottery for Micropayments

Micali and Rivest [56] suggested the following elegant way to perform non-interactive lottery (with the main application in micropayments). The merchant published a public key PK for a VRF f, the user chooses a ticket x, and wins if some predicate about $f(x)$ is true (for example, if $f(x)$ is less than some threshold t). Since f looks random to the user, the user cannot significantly bias his odds no matter what x he chooses. Similarly, since the merchant is committed to f by the public key PK, they merchant cannot lie about the value $f(x)$. Unfortunately, this still leaves exactly the same problem we had for the NIZK application above. Nothing stops the merchant from publishing a "non-balanced" VRF. In the extreme case, a constant function $f(x) = c$, where c is selected so that the predicate does not hold. Once again, we need balancedness to ensure that the merchant not only cannot change the value of f after the commitment, but also guarantees that the value $f(x)$ is random at least for a *random* x. Once again, VRPs perfectly solve this problem.

Moreover, VRPs have an extra advantage that one can *precisely* know the number of possible winners: it is exactly equal to the number of strings y satisfying the given predicate. Thus, one can always allocate a given number of prizes and never worry that with some small probability there will be more winners than prizes.

Reusable Coin-Flipping

We can extend the previous lottery example to the following coin flipping problem. Alice wants to publish some value PK (keeping the corresponding value SK secret) allowing other to non-interactively select a random number r as follows. Any party Bob can choose a random value x and send it to Alice. The value x (combined with PK) uniquely defines the final value of r. If needed, Alice can open the value of r and convince Bob that this value is correct. Additionally, we want the following properties.

(a) No matter how Bob selects x, the value r looks random to Bob (except if he "replays" some old r).

(b) For any x, Alice cannot produce two different r as the final value, even if she adversarially chooses the public key PK.

(c) Bob is sure that that if he selects x at random, the value r is random, even if Alice adversarially chooses the public key PK.

(d) Alice can reuse the same PK for many executions (and only has to worry about the replay attack from Bob).

It is clear that VRPs precisely solve this problem. In contrast, VRFs do not satisfy property (c), while other existing coin-flipping protocols are either inefficient or do not appear to have the reusability property (d).

Adding Verifiability to PRP Applications

Finally, we mention examples how VRPs could be useful to add verifiability to some application of PRPs (where, again, PRFs are not sufficient).

Verifiable CBC Encryption. As the simplest example, using VRPs one can add verifiability to CBC encryption and decryption.

Verifiable Huge Random Objects. A bit less straightforwardly, we consider the question of "truthfully", yet efficiently, sampling huge (pseudo) random objects, initiated by Goldreich et al. [36]. In this work, the authors showed several applications where PRPs can be used to efficiently sample various exponential-sized objects (like random connected graphs). Using VRPs one can naturally add verifiability to these constructs, so that the sampler can compactly commit and selectively reveal small parts of the huge object (like an edge). However, there is a subtlety. Since the PRP is often used as only part of the sampling procedure, revealing the proofs might leak a lot of extra information which might be undesirable. For example, in the random connected graph example one first samples a (pseudo) random graph, and then uses the PRP to add a random Hamiltonian cycle to it (in order to make it connected). With VRPs in place of PRPs, revealing the VRP proof will reveal that a given edge is part of the "special" Hamiltonian cycle, which is probably undesirable.

Nevertheless, we can avoid this "privacy problem" in scenarios where only PRPs are used to sample the given object. We give one such example (not present in [36]). Specifically, we can use PRPs to sample a pseudorandom constant-degree graph of exponential size (which is very likely to be a great expander). In the case the graph should be bipartite, such sampling simply consists of choosing d independent PRPs, where d is the required degree. This allows one to easily find all the neighbors of a given node on either side of the graph. In case of regular graphs, we need to sample d random matchings, which can also be done using PRPs by using an elegant result of Naor and Reingold [59] allowing one to sample pseudorandom permutations with a prescribed cycle structure. In either case, by using VRPs in place of PRPs we get *verifiable* random, constant-degree graphs, which do not suffer from the problem we had for random connected graphs.

Notice also that PRFs/VRFs are not sufficient for this application, since with high probability they will not result in a truthful implementation. Additionally, such sampling is not "reversible" (i.e., if $f(x) = y$, then given x one can see that y is connected to it, but not vice versa).

We hope that more "verifiable" huge random objects could be "privately" sampled using our technique.

Chapter 7

Relation Between the Ideal Cipher and Random Oracle Models

In the introduction, we discussed the notion of idealized models and how these make the task of designing practical and efficient protocols easier, at the cost of formally provable security in the standard model. Two of the most popular examples of idealized models are the *Random Oracle Model* (ROM) and the *Ideal Cipher Model*. In chapter 3, we discussed the problem of instantiating the *random oracle* with an actual hash function and the assumptions involved therein. We discussed the ROM in some detail there and gave *indifferentiable constructions* of the random oracle from a fixed-length input random function oracle or an ideal cipher oracle. We start by giving a short description of the *Ideal Cipher Model*.

IDEAL CIPHER MODEL. The *Ideal Cipher Model* (ICM) (also known as the “Shannon Model”) is an example of a ideal assumption model, just as the ROM. In this model, we assume the existence of a publicly accessible Ideal Block Cipher. This is essentially 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 an actual block cipher, such as AES. Although the ICM is not as popular as the ROM, there are still several examples of schemes where this model has been used [15, 23, 30, 42, 46].

Several questions have been raised regarding security in the ICM. Existing bock ciphers, 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 ROM, uninstantiable schemes that are secure in the ICM have also been discovered (see [14]). 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 TWO MODELS. We discussed the *indifferentiability framework* [52] earlier in this thesis, as a framework for comparing two ideal assumptions such as the random oracle and the ideal cipher assumptions. In particular, we used this framework to find (indifferentiable) constructions of the random oracle using an ideal cipher. The existence of such constructions implies that the random oracle assumption is no stronger an assumption than assuming the existence of an ideal cipher. That is, any cryptographic task that can be (efficiently) accomplished in the Random Oracle Model is also (efficiently) achievable in the Ideal Cipher Model using one of these indifferentiable RO constructions. Thus, the indifferentiability turns out to be the right notion when comparing two ideal assumption models.

We know from chapter 3 that there exist indifferentiable constructions of the Random Oracle using the Ideal Cipher. Thus it is really interesting to investigate the other direction of this question. That is, *Is there an indifferentiable construction of an Ideal Cipher using a Random Oracle?* This direction seems much more difficult to tackle. Actually, it is widely believed that a positive answer holds in this direction too [20]. In fact, it is conjectured that, with a sufficient number of rounds, the *Luby-Rackoff (LR) construction* [47] (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 ROM [1]. In spite of this, there has not been much progress in getting a formal proof of this conjecture.

In this chapter, we take a first step toward resolving this problem.

[1]As we already discussed in chapter 5, the LR result [47] is not applicable in this case because an attacker can easily find out all intermediate round values by querying the random oracle.

7.0.5 Our Plan

We will start by describing the notion of *indifferentiability in the honest-but-curious model*. This is a weaker notion than general indifferentiability which we described in chapter 2, but is considerably stronger than the classical notion of indistinguishability (see later). 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 consists of 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 block cipher key 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 round number). We conjecture that the LR-construction is indifferentiable from a *random permutation*, with a sufficient number of rounds. However, we will not be able to prove this result in general. Instead, we prove this implication in the honest-but-curious model, *as long as the number of rounds is super-logarithmic in the security parameter* λ. We will derive this as a consequence of lemma 20 from chapter 5.

We conjecture that our result is sub-optimal in the sense that the LR construction seems to be secure even with a "large enough" *constant* number of rounds (see later for what large enough could be), and even in the malicious model. However, we show 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 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.

7.1 Indifferentiability in the Honest-but-Curious Model

We briefly recall the general notion of indifferentiability. For two ideal primitives $\mathcal{F}$ and $\mathcal{G}$, an efficient oracle machine $C_{\mathcal{G}}$ is an *indifferentiable construction* of the ideal primitive $\mathcal{G}$ using the ideal primitive $\mathcal{F}$ if there exists a simulator $S_{\mathcal{F}}$ such that for any efficient distinguisher D, the following probability is negligible:

$$\left|\Pr[D^{C_{\mathcal{G}}^{\mathcal{F}},\mathcal{F}}(1^\lambda)=1]-\Pr[D^{\mathcal{G},S_{\mathcal{F}}^{\mathcal{G}}}(1^\lambda)=1]\right|$$

Roughly speaking, the task of the simulator $S_{\mathcal{F}}$ is to simulate the role played by the ideal primitive $\mathcal{F}$ in the $\mathcal{F}$ ideal model (from the view of the distinguisher), in the $\mathcal{G}$ ideal model. In the new (weaker) notion of *indifferentiability in the honest-but-curious model*, the distinguisher effectively has 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 oracle access to the queries made by the construction $C_{\mathcal{G}}$ to the $\mathcal{F}$ oracle, which we 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 general indifferentiability to trying to simulate the communication transcript $\mathcal{T}_{C_{\mathcal{G}\leftrightarrow\mathcal{F}}}$ in the $\mathcal{G}$ model. When the distinguisher D is in $\mathcal{F}$ model, then the queries in $\mathcal{T}_{C_{\mathcal{G}\leftrightarrow\mathcal{F}}}$ can be divided into two categories. Those for which D does not observe the queries of $C_{\mathcal{G}}$, and those for which it does. In the $\mathcal{G}$ model, the former queries are sent directly to the $\mathcal{G}$ oracle and the responses of $\mathcal{G}$ are sent back to D. 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 the output of $\mathcal{G}$ to D, it also sends a simulated communication transcript $\mathcal{T}_S$. These two views of the distinguisher are depicted in figure 7.1.

Definition 23. *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.*

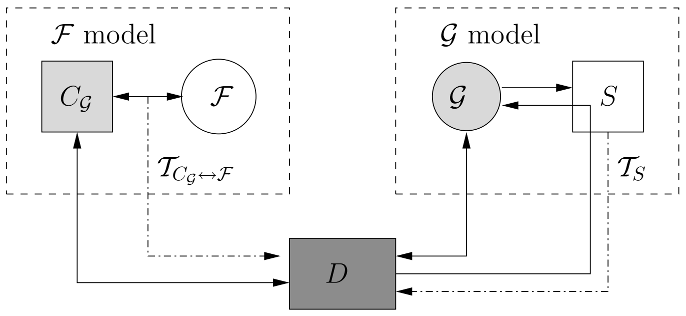


Figure 7.1: Indifferentiability in honest-but-curious model: The distinguisher D either interacts with $C_{\mathcal{G}}$ and gets the transcript $\mathcal{T}_{C_{\mathcal{G}} \leftrightarrow \mathcal{F}}$ or it interacts with $\mathcal{G}$ and gets the simulated transcript $\mathcal{T}_S$

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 the ideal $\mathcal{G}$ oracle.

Consider the construction $C_{\mathcal{G}}$ that is 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 24. *If a construction $C_{\mathcal{G}}$ using $\mathcal{F}$ is indifferentiable from $\mathcal{G}$ in the honest-but-curious model, as stated in definition 23, 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 7.2] 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 23 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

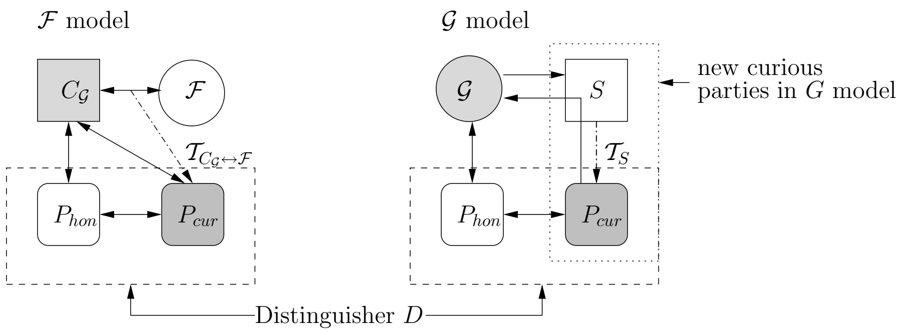


Figure 7.2: An idea of the proof of lemma 24. The conventional honest parties P_{hon} along with the curious ones P_{cur} can be seen together as a distinguisher D

honest 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. 23) simply by ignoring the transcripts $\mathcal{T}_{C_{\mathcal{G}} \leftrightarrow \mathcal{F}}$ (or $\mathcal{T}_S$).

7.1.1 Transparent Constructions

Even though general indifferentiability seems to be much stronger than indifferentiability in the honest-but-curious model (definition 23), we now show that for certain types of constructions these two definitions are, in fact, equivalent.

Definition 24 (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 25. *If a transparent construction $C_{\mathcal{G}}$ (using $\mathcal{F}$) is (polynomially) indifferentiable from $\mathcal{G}$ in the honest-but-curious model (defn. 23) then it is also (polynomially) indifferentiable from $\mathcal{G}$.*

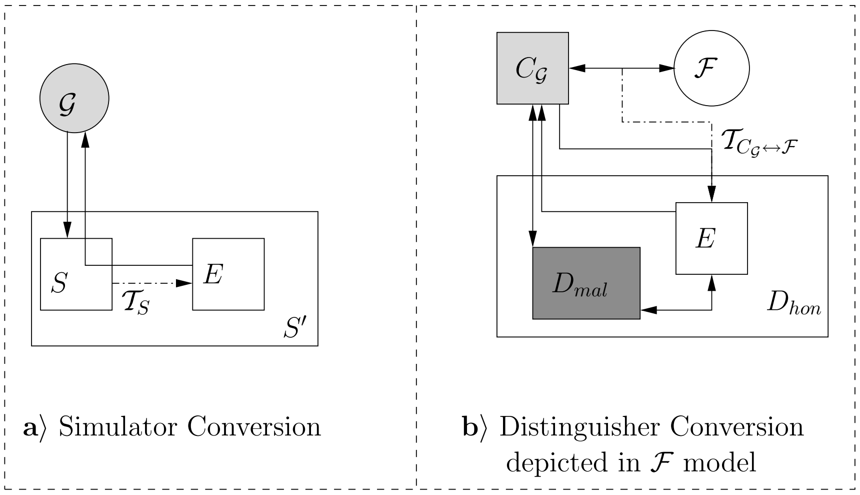
Proof: Say that a construction $C_{\mathcal{G}}$ is indifferentiable from ideal primitive $\mathcal{G}$ in the honest-but-curious model. Then we have a simulator S_{hon} that successfully fakes the transcript $\mathcal{T}_{C_{\mathcal{G}} \leftrightarrow \mathcal{F}}$ (with $\mathcal{T}_{S_{hon}}$) in 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 23 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 7.3a.

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 7.3b.

Say $C_{\mathcal{G}}$ 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 $C_{\mathcal{G}}$.



$\mathcal{G}$
S
E
$\mathcal{T}_S$
S'
$C_{\mathcal{G}}$
$\mathcal{F}$
$\mathcal{T}_{C_{\mathcal{G}} \leftrightarrow \mathcal{F}}$
E
D_{mal}
D_{hon}
a⟩ Simulator Conversion
b⟩ Distinguisher Conversion depicted in $\mathcal{F}$ model

Figure 7.3: **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}.

7.2 The Construction

In this section, we will propose a construction for 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 suffices to give a construction C_π of a single random permutation (RP) $\pi : \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$ using H. Similar to the ideal cipher oracle, the random permutation oracle π responds to both forward and inverse queries. On input $(0, x)$, it outputs $y = \pi(x)$ and on input $(1, y)$, it outputs x such that $\pi(x) = y$. A constriction for the ideal cipher can be derived from this RP construction by prepending the ideal cipher key to every query C_π makes to H.

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

The Random Permutation Construction. We first note that the constructions in [47, 58] 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) $H : \{0,1\}^* \rightarrow \{0,1\}^n$. Similar to [47, 58], 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 25 (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 $\pi : \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$. 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 $H : \{0,1\}^* \rightarrow \{0,1\}^n$) is an indifferentiable construction of a random permutation $\pi : \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$.*

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.

7.2.1 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 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 33. *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: We need to design an extracting algorithm Ext that when given access to $(C_{\pi,k}, \mathcal{T}_{C_{\pi,k} \leftrightarrow H})$ can extract the values of $H(\langle i \rangle \parallel x)$ for any $x \in \{0,1\}^n$ and $i \in \{1 \ldots k\}$. We will also refer to the function output $H(\langle i \rangle \parallel x)$ as $h_i(x)$.

The proof of this theorem is similar to the theorem 18 in chapter 5. In particular, we will use the algorithm E (described there) that takes as input a round number j and a $2n$ bit string Y, and finds the input such that Y forms the j and $(j+1)$.

The extractor Ext gets as input $\langle i \rangle$ and x. It runs the algorithm E on input $((i-1), x' \parallel x)$, for an arbitrary n bit string x'. It responds to the queries made by E using the construction $C_{\pi,k}$ and can provide all intermediate round values from the transcript $\mathcal{T}_{C_{\pi,k} \leftrightarrow H}$. Upon finding this input X, the extractor Ext simply sends this as a query to $C_{\pi,k}$ and learns the output $h_i(x)$ from the transcript $\mathcal{T}_{C_{\pi,k} \leftrightarrow H}$.

This extractor Ext makes $\mathcal{O}(\mathsf{Fibonacci}(k))$ number of queries (and runs in time $\mathcal{O}(\mathsf{Fibonacci}(k))$ as well), just like E. For number of rounds $\mathcal{O}(\log \lambda)$, this is polynomial in the security parameter λ. ◻

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.

7.2.2 HBC Indifferentiability 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 34. *The k round construction $\mathcal{C}_{\pi,k}$ is $(t_D,\ t_S,\ q,\ \mathcal{O}\left((q \cdot k)^5 \cdot 2^{-n}\right)\ )$ 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: The proof of this theorem consists of two parts: first, we will describe the simulator S that fakes the communication between the construction $C_{\pi,k}$ and H in the random permutation model, and next we will give a proof of indifferentiability (in HBC model) using this simulator.

THE SIMULATOR. The simulator S gets inputs either of the form $(0, x)$ (forward queries) or of the form $(1, y)$ (inverse queries), where $x, y \in \{0,1\}^{2n}$. In the random oracle model, if the input $(0, x)$ is given to the construction $C_{\pi,k}$, then $C_{\pi,k}$ makes queries to the random oracle H and computes the round values $R_0 \ldots R_{k+1}$ 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 are handled in a similar fashion, albeit in reverse, starting from $R_k = y|_{L}, R_{k+1} = y|_{R}$ 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 round function h_i, in which it stores all previously generate round function outputs for h_i. Consider a forward query $(0, x)$, thus $R_0 = x|_L$ and $R_1 = x|_R$. The simulator generates a fake transcript for this query as follows:

1. It forwards the 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 $[R_{i-1} = h_i(R_i) \oplus R_{i+1}]_{i=k\ldots bot}$, where $bot \in \{1 \ldots k\}$. If $bot = 1$, then all the round values for this query are already defined, so it checks to see if $(R_{bot-1} \parallel R_{bot}) = x$. If so, S returns this sequence of round value/round function output pairs 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 sequence of defined round values going down from $R_0 = x|_L$ and $R_1 = x|_R$. That is, a sequence of round values $[R_{i+1} = h_i(R_i) \oplus R_{i-1}]_{i=1\ldots top}$, where $top \in \{1 \ldots k\}$. 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 the previously generated fake transcript.

4. If everything goes well until this point, then the simulator S starts defining the missing round function outputs between *top* and *bot*. It

defines the function outputs $h_{top+1}(R_{top+1}) \dots h_{bot-2}(R_{bot-2})$ at random. It connects the top and bottom sequences of round values 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 simply tries to define all round function values randomly. However, it first scans to see if some of the intermediate round values were already defined in its previous responses. It does so both starting from top and bottom, and defines the undefined round function outputs in the middle at random but making sure that it connects the two partial sequences of round values. If it so happens that there are no undefined round values in the middle, then it realizes that it cannot be consistent with both these partial sequences of defined round values simultaneously and exits with failure.

Proof of HBC Indifferentiability. Now we will prove that when the simulator S described above is used in the indifferentiability game, then any distinguisher D that makes at most (a polynomial) 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 7.4a). Since we have made no real change from the point of view of the distinguisher, we have $Pr[D^{(\pi,\mathcal{T}_{S^{\pi}})} = 1] = Pr[D^{(\mathcal{M}^{\pi},\mathcal{T}_{S^{\pi}})} = 1]$.

<u>Bounding out the "bad events":</u> 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 $h_1(R_1)\ldots h_k(R_k)$ 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 $R_0^{(m)}, R_1^{(m)}, \ldots, R_k^{(m)}, R_{k+1}^{(m)}$.

When assigning a new round function value $h_i(R_i^{(m)})$ for query number m, the new simulator S_1 makes sure that the new round round value generated, i.e. $R_{i+1}^{(m)}$ (resp. $R_{i-1}^{(m)}$) if the m^{th} query is a forward query (resp. inverse query), cannot be represented as an *XOR of upto five previously existing round values.* That is, the simulator S_1 intentionally prevents a *5-XOR condition* (see chapter 5) from occurring in its responses.

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 results in the *5-XOR condition* being true. 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, $Pr\left[B_1\right] = \mathcal{O}\left(\frac{(q \cdot k)^6}{2^n}\right)$. This can be derived by using the birthday paradox to bound the probability that any XOR of upto 6 round values is 0^n.

<u>Transferring Control to the Simulator:</u> 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 7.4b).

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 recover $\pi(x)$ from $\mathcal{T}_{S_1}$ and respond 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 26. *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: In fact, this lemma can be seen as a consequence of the combinatorial lemma 20 from chapter 5. In order to see this, consider the situation in which the simulator S_1 *exits with failure.* This occurs if there exist partial sequences of round values $R_0^{(m)}, R_1^{(m)}, \ldots, R_{top}^{(m)}, R_{top+1}^{(m)}$ 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 $top \geq (k/2)$ or $bot \leq (k/2) + 1$. Thus, we can deduce that at least one of the round function outputs $h_{k/2}(R_{k/2}^{(m)}$ or $h_{k/2+1}(R_{k/2+1}^{(m)}$ is already defined. This can only occur if the corresponding round value in the m^{th} query collides with the corresponding round value in an earlier query. Moreover, this earlier query must be different from the m^{th} query, otherwise the simulator S_1 would not have been inconsistent in the m^{th} query. Hence, the $(k/2)^{th}$ or the $((k/2) + 1)^{th}$ round value collides for two of the queries made by the distinguisher. But since the simulator S_1 makes sure that the *5-XOR condition* does not hold, we can deduce that the number of queries q made by the distinguisher must be exponential in the security parameter, i.e. $q = \mathcal{O}(1.3803^k)$. $\square$

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

<u>**Removing the Random Permutation π:**</u> Until now, all responses of the simulator 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 7.4c), so that the actual permutation output may not be consistent with π.

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 26 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,

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

<u>Onto the Random Oracle Model:</u> 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 7.4d), we have $Pr[D^{(\mathcal{M}^{\mathcal{T}_{S_2}},\mathcal{T}_{S_2})}] = Pr[D^{(\mathcal{C}^H_{\pi,k},\mathcal{T}_{C_{\pi,k}\leftrightarrow H})} = 1]$.

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

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

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 $\mathcal{O}\left(\frac{q^4}{2^n}\right)$ $\square$

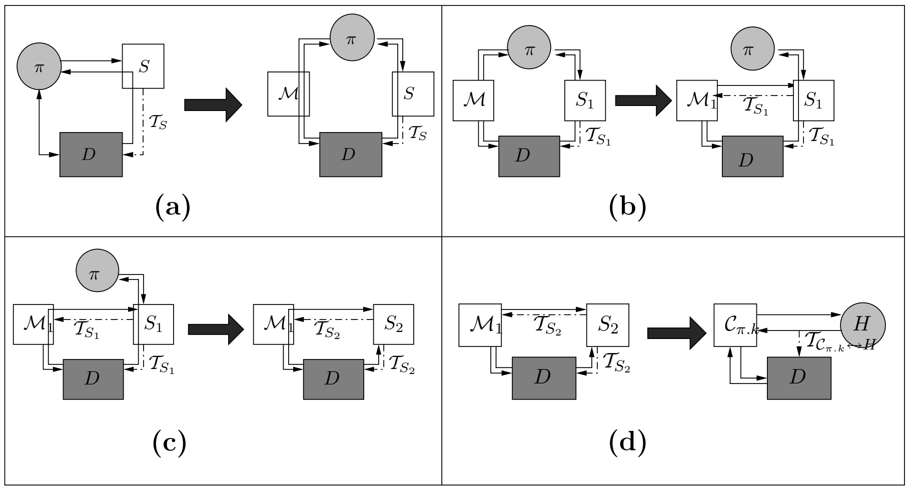


Figure 7.4: Overall Game Structure

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

One can deduce from theorem 34 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 35. *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: This theorem can also be derived as a consequence of the lemma 26. In particular, if there exists an extracting algorithm Ext that can compute $h_i(x)$, given as input $\langle i\rangle \parallel x$, then it cannot be efficient for number of rounds $k = \omega(\log\lambda)$. In particular, if Ext works for the k round LR-construction $\mathcal{C}_{\pi,k}$ with the actual round values, then it should also work for the hybrid scenario in figure 7.4b, where it has oracle access to π and the round values are faked by the simulator S_1 (that avoids the 5-XOR condition).

However, in this case, we can show that if Ext is efficient, then we can construct another (efficient) algorithm A that finds two queries that collide in the $(k/2)^{th}$ round value. In particular, A chooses a *random input* $X \in \{0,1\}^{2n}$ and makes this query to π (and gets all intermediate round values from S_1). It then runs the extracting algorithm Ext with input $\langle k/2 \rangle \parallel R_{k/2}$ (where $R_{k/2}$ is the round value that it gets from the faked transcript of S_1). Since Ext is extremely unlikely to guess the input X used by A, it will find out $h_{k/2}(R_{k/2})$ through a permutation query to π that is different from X, with overwhelming probability. The collision finding algorithm A can get this different input by keeping track of the queries made by Ext, and thus find two queries that collide in the $(k/2)^{th}$ round value. However, this is impossible from the combinatorial lemma 20 from chapter 5, since S_1 prevents the 5-XOR condition from holding. $\square$

7.2.4 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 [20] showed that for less than 6 rounds the LR-construction is not indifferentiable from a random permutation.

Theorem 36 ([20]). *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.*

Proof: 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 [47]. Coron [20] gave attacks on 4 and 5 round LR-constructions in the indifferentiability scenario. We give an attack on the 4 round LR construction here for illustration.

Let us represent the round values of the construction $C_{\pi,4}$ as $R_0, R_1 \ldots R_4, R_5$, such that $C_{\pi,4}(R_0 \parallel R_1) = (R_4 \parallel R_5)$. 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: $(C_{\pi,4}, H)$ 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 $C_{\pi,4}$. 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 $C_{\pi,4}$. 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. $\square$

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 [47]. Thus we can derive the following corollary from theorem 36.

Corollary 3. *A* 4 *round LR-construction is indistinguishable , but not indifferentiable, from a random permutation (even in the honest-but-curious model).*

Bibliography

[1] J. H. An, M. Bellare, *Constructing VIL-MACs from FIL-MACs: Message Authentication under Weakened Assumptions*, CRYPTO 1999, pages 252-269.

[2] B. Barak, *How to Go Beyond the Black-Box Simulation Barrier?*, Proc. of 42nd FOCS, pp. 106-115, 2001.

[3] M. Bellare, *New Proofs for NMAC and HMAC: Security without Collision-Resistance*, Advances in Cryptology - Crypto 2006 Proceedings, Springer-Verlag, 2006.

[4] Mihir Bellare, Alexandra Boldyreva and Adriana Palacio. An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem. Proccedings of Eurocrypt 2004.

[5] M. Bellare, R. Canetti, and H. Krawczyk, *Pseudorandom Functions Revisited: The Cascade Construction and Its Concrete Security*, In Proc. 37th FOCS, pages 514-523. IEEE, 1996.

[6] M. Bellare, R. Canetti, and H. Krawczyk, *Keying hash functions for message authentication*, Advances in Cryptology - Crypto 96 Proceedings, LNCS Vol. 1109, Springer-Verlag, 1996.

[7] M. Bellare, J. Kilian, and P. Rogaway. The Security of Cipher Block Chaining. In *Crypto '94*, pages 341–358, 1994. LNCS No. 839.

[8] M. Bellare and P. Rogaway, *Random oracles are practical : a paradigm for designing efficient protocols*. Proceedings of the First Annual Conference on Computer and Commmunications Security, ACM, 1993.

[9] M. Bellare and P. Rogaway, *Optimal Asymmetric Encryption*, Proceedings of Eurocrypt'94, LNCS vol. 950, Springer-Verlag, 1994, pp. 92–111.

[10] 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.

[11] M. Bellare and P. Rogaway, *Collision-Resistant Hashing: Towards Making UOWHFs Practical*, In *Crypto '97*, LNCS Vol. 1294.

[12] M. Bellare and T. Ristenpart, *Multi-Property-Preserving Hash Domain Extension and the EMD Transform*, In Advances in Cryptology - Asiacrypt 2006.

[13] M. Bellare and T. Ristenpart, *Hash Functions in the Dedicated-Key Setting: Design Choices and MPP Transforms*, in ICALP 2007.

[14] J. Black, *The Ideal-Cipher Model, Revisited: An Uninstantiable Blockcipher-Based Hash Function*, *eprint 2005/210*, (2005).

[15] 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.

[16] Manuel Blum, *Coin Flipping by Telephone - A Protocol for Solving Impossible Problems*, COMPCON 1982: 133-137.

[17] R. Canetti, *Universally Composable Security: A New Paradigm for Cryptographic Protocols*, proceedings of the 42nd Symposium on Foundations of Computer Science (FOCS), 2001. Cryptology ePrint Archive, Report 2000/067, `http://eprint.iacr.org/`.

[18] R. Canetti, O. Goldreich and S. Halevi, *The random oracle methodology, revisited*, STOC' 98, ACM, 1998.

[19] Ran Canetti, Oded Goldreich and Shai Halevi. On the random oracle methodology as applied to Length-Restricted Signature Schemes. In *Proceedings of Theory of Cryptology Conference*, pp. 40–57, 2004.

[20] J.-S. Coron, *personal communication*, 2002.

[21] J.-S. Coron, Y. Dodis, C. Malinaud and P. Puniya, *Merkle-Damgård Revisited: How to Construct a Hash Function*, Advances in Cryptology, Crypto 2005 Proceedings: 430-448, Springer-Verlag, 2006.

[22] I. Damgård, *A Design Principle for Hash Functions*, In Crypto '89, pages 416-427, 1989. LNCS No. 435.

[23] Anand Desai, *The Security of All-or-Nothing Encryption: Protecting against Exhaustive Key Search*, CRYPTO 2000: 359-375.

[24] Y. Dodis, *Efficient construction of (distributed) verifiable random functions*, In *Proceedings of 6th International Workshop on Theory and Practice in Public Key Cryptography*, pp 1 -17, 2003.

[25] Y. Dodis, R. Gennaro, J. Håstad, H. Krawczyk, and T. Rabin, *Randomness Extraction and Key Derivation Using the CBC, Cascade and HMAC Modes*, Advances in Cryptology - CRYPTO, August 2004.

[26] Y. Dodis, R. Oliveira, K. Pietrzak, *On the Generic Insecurity of the Full Domain Hash*, Advances in Cryptology - CRYPTO, August 2005.

[27] Yevgeniy Dodis and Prashant Puniya, *Feistel Networks made Public, and Applications*, Advances in Cryptology - EUROCRYPT, May 2007.

[28] Y. Dodis and A. Yampolskiy, *A Verifiable Random Function With Short Proofs and Keys*, In *Workshop on Public Key Cryptography (PKC)*, January 2005.

[29] C. Dwork and M. Naor, *Zaps and their applications*, In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pp 283, 2000.

[30] Shimon Even and Yishay Mansour, *A Construction of a Cipher From a Single Pseudorandom Permutation*, ASIACRYPT 1991: 210-224.

[31] Uriel Feige, Dror Lapidot and Adi Shamir, *Multiple NonInteractive Zero Knowledge Proofs Under General Assumptions*, in *SIAM Journal of Computing* 29(1): 1-28 (1999).

[32] FIPS 180-1, *Secure hash standard*, Federal Information Processing Standards Publication 180-1, U.S. Department of Commerce/N.I.S.T., National Technical Information Service, Springfield, Virginia, April 17 1995 (supersedes FIPS PUB 180).

[33] National Institute of Standards and Technology (NIST). Secure hash standard. FIPS 180-2. August 2002.

[34] RFC 1321, *The MD5 message-digest algorithm*, Internet Request for Comments 1321, R.L. Rivest, April 1992.

[35] O. Goldreich, S. Goldwasser and S. Micali, *How to Construct Random Functions* in *Journal of the ACM*, Vol. 33, No. 4, October 1986.

[36] Oded Goldreich, Shafi Goldwasser and Asaf Nussboim, *On the Implementation of Huge Random Objects*, FOCS 2003: 68-79.

[37] Oded Goldreich and Leonid A. Levin, *A Hard-Core Predicate for all One-Way Functions*, STOC 1989: 25-32.

[38] Shafi Goldwasser and Rafail Ostrovsky, *Invariant Signatures and Non-Interactive Zero-Knowledge Proofs are Equivalent (Extended Abstract)*, in *CRYPTO 1992*: 228-245.

[39] Shafi Goldwasser and Yael 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.

[40] S. Halevi and H. Krawczyk, *Strengthening Digital Signatures Via Randomized Hashing*, in Advances in Cryptology - CRYPTO 2006, pp. 41-59.

[41] H. Handschuh and D. Naccache, *SHACAL*, In B. Preneel, Ed., First Open NESSIE Workshop, Leuven, Belgium, November 13-14, 2000.

[42] 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.

[43] Antoine Joux, *Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions*, Advances in Cryptology - CRYPTO 2004, 306-316.

[44] J. Kelsey, in CRYPTO 2005 Rump Session.

[45] J. Kelsey and T. Kohno, *Herding Hash Functions and the Nostradamus Attack*, Advances in Cryptology - EUROCRYPT 2006, 183-200.

[46] 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.

[47] M. Luby and C. Rackoff, *How to construct pseudo-random permutations from pseudo-random functions*, SIAM J. Comput., Vol. 17, No. 2, April 1988.

[48] Stefan Lucks. *Design Principles for Iterated Hash Functions*, available at E-Print Archive, `http://eprint.iacr.org/2004/253`.

[49] A. Lysyanskaya, *Unique Signatures and verifiable random functions from DH-DDH assumption*, in *Proceedings of the 22nd Annual International Conference on Advances in Cryptography (CRYPTO)*, pp. 597 612, 2002.

[50] Ueli M. Maurer, Yvonne Anne Oswald, Krzysztof Pietrzak and Johan Sjödin, *Luby-Rackoff Ciphers from Weak Round Functions?*, EUROCRYPT 2006: 391-408.

[51] Ueli M. Maurer and Krzysztof Pietrzak, *The Security of Many-Round Luby-Rackoff Pseudo-Random Permutations*, in *EUROCRYPT 2003*, 544-561.

[52] 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.

[53] Ueli Maurer and Johan Sjodin. *Single-key AIL-MACs from any FIL-MAC*, In *ICALP 2005*, July 2005.

[54] R. Merkle, *One way hash functions and DES*, Advances in Cryptology, Proc. Crypto'89, LNCS 435, G. Brassard, Ed., Springer-Verlag, 1990, pp. 428-446.

[55] S. Micali, M. Rabin and S. Vadhan, *Verifiable Random functions*, In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science*, pp. 120 -130, 1999.

[56] Silvio Micali and Ronald L. Rivest, *Micropayments Revisited*, CT-RSA 2002, 149-163.

[57] Moni Naor, *Bit Commitment Using Pseudo-Randomness*, CRYPTO 1989: 128-136.

[58] Moni Naor and Omer Reingold, *On the construction of pseudo-random permutations: Luby-Rackoff revisited*, in *Journal of Cryptology*, vol 12, 1999, pp. 29-66.

[59] Moni Naor and Omer Reingold, *Constructing Pseudo-Random Permutations with a Prescribed Structure*, J. of Cryptology, vol 14, 2001.

[60] Moni Naor and Moti Yung, *Universal One-Way Hash Functions and their Cryptographic Applications*, STOC 1989: 33-43.

[61] Jesper Buus Nielsen. Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-Committing Encryption Case. In *Advances in Cryptology - Crypto 2002 Proceedings* (2002), 111 -126

[62] National Institute of Standards and Technology, *NIST's Plan for New Cryptographic Hash Functions*, http://www.csrc.nist.gov/pki/HashWorkshop/index.html.

[63] Jacques Patarin, *How to construct pseudorandom permutations from a single pseudorandom function*, *Advances in Cryptology - EUROCRYPT '92*, Lecture Notes in Computer Scence, vol.658, pp. 256-266, Springer-Verlag, 1992.

[64] Jacques Patarin, *Security of Random Feistel Schemes with 5 or More Rounds*, in *CRYPTO 2004*, 106-122.

[65] PKCS #1 v2.1, *RSA Cryptography Standard (draft)*, document available at `www.rsa security.com/rsalabs/pkcs`.

[66] B. Pfitzmann and M. Waidner, *A model for asynchronous reactive systems and its application to secure message transmission*. In IEEE Symposium on Security and Privacy, pages 184-200. IEEE Computer Society Press, 2001.

[67] J. Pieprzyk, *How to construction pseudorandom permutations from single pseudorandom functions*, *Advances in Cryptology - EUROCRYPT*

'90, Lecture Notes in Computer Scence, vol. 473, pp. 140-150, Springer-Verlag, 1990.

[68] B. Preneel, R. Govaerts and J. Vandewalle, *Hash Functions Based on Block Ciphers: A Synthetic Approach*, in Advances in Cryptology - CRYPTO '93,, Santa Barbara, California, USA.

[69] Z. Ramzan and L. Reyzin, *On the Round Security of Symmetric-Key Cryptographic Primitives*, in Advances in Cryptography - Crypto, LNCS vol. 1880, Springer-Verlag, 2000.

[70] V. Shoup, *A composition theorem for universal one-way hash functions*, In *Eurocrypt '00*, pp. 445–452, LNCS Vol. 1807.

[71] Daniel R. Simon, *Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions?*, EUROCRYPT 1998: 334-345.

[72] X. Wang, H. Yu, Y. L. Yin, *Efficient Collision Search Attacks on SHA-0*, Advances in Cryptology - CRYPTO 2005, 1-16.

[73] X. Wang, Y. L. Yin, H. Yu, *Finding Collisions in the Full SHA-1*, Advances in Cryptology - CRYPTO 2005: 17-36.

[74] 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.