On the Blackbox Reduction of some Cryptographic Constructions

A thesis presented to the Indian Statistical Institute
in fulfillment of the thesis requirement for the degree of
Doctor of Philosophy in Computer Science

by

Rishiraj Bhattacharyya

Supervisor: Professor Bimal Kumar Roy

Applied Statistics Unit
INDIAN STATISTICAL INSTITUTE
Kolkata, West Bengal, India
October, 2011

Abstract

Merriam- Webster's open dictionary defines the term black box as

"a usually complicated electronic device whose internal mechanism is usually hidden from or mysterious to the user; broadly : anything that has mysterious or unknown internal functions or mechanisms."

In the world of computer science, when we use a program as blackbox, we use it only by the input/output relation of the program. One does not bother about the particular code of the program and more specifically do not access the internal variables while executing the program.

In cryptography, blackbox constructions are very popular due to the efficiency and the robustness. As opposed to specific white box constructions, security proofs of black box constructions remain valid even if only one particular function (bilinear map) satisfies the underlying assumption (like one-wayness) and some other function (like RSA) does not.

Indifferentiability of popular domain extension algorithms In this thesis we prove blackbox reduction and separation of some popular cryptographic constructions. In symmetric key constructions, we consider blackbox reduction of cryptographic hash functions under the indifferentiability framework. We introduce a unified framework for indifferentiability security analysis by providing an indifferentiability upper bound for a non-invertible function based, wide class of hash designs, called GDE or *generalized domain extension*. In our framework, we present a unified simulator and avoid the problem of defining different simulators for different constructions. We show, the probability of some bad event (based on interaction of the attacker with the GDE and the underlying ideal primitive) is actually an upper bound for indifferentiable security. As immediate applications of our result, we provide simple and improved (in fact optimal) indifferentiability upper bounds for HAIFA and tree (with counter) mode of operations. We also consider a particular permutation based mode of operation of a SHA3 candidate, JH. In this thesis, we prove indifferentiability of JH mode of operation. Additionally, a new mode by modifying the JH mode is also analyzed in the indifferentiability framework.

In the second part of the thesis, we prove separation of popular public key constructions from a wide class of assumptions.

Generic Insecurity of PSS First, we consider the problem of securely instantiating Probabilistic Signature Scheme (PSS) in the standard model. PSS, proposed by Bellare and Rogaway is a widely deployed randomized signature scheme, provably secure (*unforgeable under adaptively chosen message*

attacks) in Random Oracle Model.
Our main result is a black-box impossibility result showing that one cannot prove unforgeability of PSS against chosen message attacks using blackbox techniques even assuming existence of *ideal trapdoor permutations* (a strong abstraction of trapdoor permutations which inherits all security properties of a random permutation, introduced by Kiltz and Pietrzak in Eurocrypt 2009) or the recently proposed *lossy trapdoor permutations*. Moreover, we show *onewayness*, the most common security property of a trapdoor permutation does not suffice to prove even the weakest security criteria, namely *unforgeability under zero message attack*. Our negative results can easily be extended to any randomized signature scheme where one can recover the random string from a valid signature.

Separation of OAEP: Optimal Asymmetric Encryption Padding (OAEP) is a widely deployed padding based encryption scheme. Although, OAEP has been shown to achieve *Indistinguishability* under Chosen Ciphertext Attack (IND-CCA), until recently, all the positive instantiation result of OAEP was on Random Oracle Model. In CRYPTO 2010, Kiltz, O'Neil and Smith proved that OAEP, based on pairwise independent hash functions and Lossy Trapdoor Permutations, can achieve weaker *Indistinguishability* under Chosen Plaintext Attack (IND-CPA) in the standard model. Achieving IND-CCA security of OAEP without random oracle remains a challenge. In Eurocrypt 2009, Kiltz and Pietrzak showed that IND-CCA security of OAEP or any other padding based encryption scheme cannot be proven IND-CCA secure from any security property satisfied by a random permutation using blackbox techniques in the standard model. In this thesis, we analyze the possibility of blackbox reduction of OAEP in the *Seed Incompressibility* model where the adversary gets oracle access to the hash functions and some fixed length advice. The Seed Incompressibility model is based on the notion of seed incompressible function, proposed by Halevi, Myers and Rackoff in TCC 2008 and can be seen as a bridge between the standard model and the random oracle model. In this thesis we formalize the notion of seed incompressibility model and extend the negative results of Kiltz and Pietrzak to this model. We show that even if the adversary can use the advice to construct only one ciphertext without querying the hash function oracles, IND-CCA security for OAEP cannot be achieved from any security property satisfied by a *random permutation* or from *Lossy Trapdoor Permutations*. Moreover, if the size of the random string is relatively small then the result is perfectly valid even when the adversary needs to query only one of the hash oracles in order to create a single ciphertext using the advice.

To Tandoori Chicken

Acknowledgement

The past three and half years at the Cryptology Research Group have been highly interesting and educative experience for me. I am indebted to my advisor, Professor Bimal Roy for being a patient and encouraging mentor. He introduced me to a lot of beautiful theories and intuition, yet gave me my own space and time to build my own ideas. Besides being an academic mentor, Bimalda has been a wonderful source of inspiration. His view towards academics and overall towards life has enabled me to pass through some very tough times.

I am thankful to Professor Rana Barua, Professor Palash Sarkar, Professor Subhamoy Maitra and Dr. Kishan Chand Gupta for their constant support and encouragement. I am indebted to Palashda as he always made time for discussion for every little doubt I had. He also took a lot of pain to make sure that I get suitable research environment.

It was a delight to work with my collaborators during my PhD; Avradip, Mridulda and Sourav. I learned a lot of things from them and looking forward to many fruitful collaborations in the future.

In the last five years at ISI, I have made some wonderful friends. I want to thank Sumit for being so patient and encouraging. I want to thank Sourav Sen Gupta for all his supports and encouragements. I wish I could be so talented yet so humble like him. I want to thank past and present members of CRG who made these years cheerful and fruitful starting from technical discussions to eating out. A big thank you to Premda, Sumantada, Sushmitadi, Somitrada, Mahabirda, Santanu, Arnab, Srimantada, Sanjay, Butu, Somindu, Sashank, Pratyay.

During my PhD I have visited Kyushu University, Korea University, INRIA Paris Rocquencourt, University of Luxembourg and Ruhr University Bochum. I sincerely thank all my hosts for those unforgettable times.

Finally, I want to thank my family for their support and understanding. They are the reason I could do what I really want to do.

Contents

List of Figures

Chapter 1

Introduction

1.1 Blackbox Reduction

The concept of *subroutine* is one of the fundamental concepts in Computer Science. A subroutine can be viewed as a string of symbols which implements a particular function. For example, consider the following piece of code written in C language.

```
int f ( int x) {
    x++;
    return x;
}
```

Essentially the subroutine f implements a function $f(.)$, where $f(x) = x+1$, for any integer x. Now consider the following piece of program that has become the symbol of the C programming language.

```
int main ( ) {
    printf ("hello world");
    return 0;
}
```

The "main" program *calls* a subroutine "printf(.)" which prints a string at the screen. Although, all C programmers use it numerous times, only few know what exactly are the instructions that "printf" executes. Indeed the operation of the program "main" does not depend upon the actual description of the subroutine "printf", rather the subroutine is used only to perform the function it is supposed to do. The program uses the subroutine "printf" as a black-box to print a string on the screen so that it can feed it some input and, in some sense, receive an output. We call such an algorithm which uses the subroutines as a black-box, a *black-box* algorithm.

Black-box algorithms are very popular in computer science. Many times, when trying to solve a particular task, one would write an algorithm that solves the task if it is given black-box access to programs that solve some simpler tasks, and then write programs that solve these simpler tasks. This approach is a basic paradigm of software engineering, and almost all programming languages implement mechanisms (such as function calls) to facilitate it. Black-box algorithms are very popular also in theoretical aspects of Computer Science. For example, when proving that a decision problem L is NP-complete, one shows the existence of a black-box algorithm B, such that if B is given black-box access to an algorithm A that solves the problem L, then B can solve the Satisfiability problem.

The reason that black-box algorithms are so popular is that it seems very hard to make use of the particular representation of a program as a string. Understanding the properties of a function from the code of a program that computes it (also known as reverse-engineering) is a notoriously hard problem. In fact, programs written without any effort on readability or programs written or compiled to low level languages are considered to be quite incomprehensible. Moreover for the problems like the Halting problem or the Satisfiability problem, it is either proven or widely believed that when trying to learn properties of a function, there is no significant advantage to use the code of the program, over getting black-box access to it.

1.1.1 Blackbox Reduction in Cryptography

In a large body of work in Cryptography, cryptographic constructions are not shown to be unconditionally secure. Rather, their security is reduced to the security of seemingly weaker or simpler primitives. It is known that if one way functions exist, then private-key encryption and authentication, pseudorandom generators, (public-key) digital signatures, zero-knowledge proofs etc are possible. On the other hand, if one way functions do not exist, none of the above problems have any solution.

Several other cryptographic problems, like public key cryptography, oblivious transfer, collision resistant hash functions etc. are not known to be equivalent to the existence of one-way functions. Moreover known constructions of digital signature or pseudorandom generators using one-way functions are inefficient. It is natural to ask whether one way function can be used to solve the problems like public key cryptography or whether one can find an efficient constructions of digital signature from one-way function. However, one has to be careful while formalizing such questions. Although, Oblivious transfer can be solved using some widely believed number theoretic assumptions, such a result does not imply a solution using arbitrary one-way function.

Impagliazzo and Rudich formalized a framework to solve such problems. They observed that most implications in cryptography are proved using a blackbox reduction, where the underlying primitive is treated as an *oracle* subroutine. Informally, a black-box reduction of a primitive P to a primitive Q is a construction of P, out of Q, that ignores the internal structure of the implementation of Q and uses it only as a black-box. In addition, in the case of fully-black-box reductions, the proof of security (showing that an adversary that breaks the implementation of P implies an adversary that breaks the implementation of Q), is black-box as well. The internal structure of the adversary that breaks the implementation of P is ignored.

1.2 Goal of the Thesis

The main objective of the thesis is to find out whether some popular cryptographic constructions can be reduced to ideal primitives using black-box reduction. We consider constructions from Symmetric Key as well as Public Key settings. For symmetric key constructions, we analyze wide class of domain extension techniques of cryptographic hash functions and check whether security notion like *Pseudo-randomness* can be achieved from the randomness of the underlying primitive (compression function or permutation) using black box techniques. For public key cryptography, our goal is to investigate whether one can use blackbox reduction to prove the security of popular padding based signature schemes (e.g. PSS) and padding based encryption schemes (e.g. OAEP) when the underlying hash function is not modeled as a Random Oracle .

1.3 Contribution of the Thesis

Indifferentiability of Popular Mode of Operations

In the first part of this thesis we consider the reduction of Pseudorandomness of a Mode of operation from the randomness of the underlying primitives in the indifferentiability framework. Although many popular domain extensions algorithms based on ideal compression functions have been shown to be indifferentiable from a Random Oracle (RO), the proof of these results are usually complicated (many times, due to numerous game hopings and hybrid arguments). Also, they require different simulators for each individual hash design. There were no known sufficient conditions for hash functions to be indifferentiable from an RO. So a natural question to ask is: *Can we characterize the minimal conditions of a cryptographic hash function to be indifferentiable from a Random Oracle and achieve optimal bound?*

In this thesis, we present a unified technique of proving indifferentiable security for a major class of iterated hash functions, called Generalized Domain Extensions. We extend the technique of Maurer for indistinguishability of random systems to the indifferentiability framework. We identify a set of events (called BAD events) and show that any distinguisher, even with unbounded computational power, has to provoke the BAD events in order to distinguish the hash function C from a random function R. Moreover we prove that, to argue indifferentiability of a construction C^f (based on compression function f), one has only to show that the probability that any distinguisher invokes those BAD events, while interacting with the pair (C^f, f), is negligible. Then, we apply our technique to some popular domain extension algorithms to provide optimal indifferentiable bounds. In particular, we consider Merkle-Damgård with HAIFA and tree mode with specific counter scheme. Many of candidates of *SHA3* competition, most notably BLAKE, MD6 etc, actually used these two modes of operations. So, our result can also be viewed as an **optimal** indifferentiability bounds of these candidates. Next, we consider blackbox reduction of the modes of operation based on permutation. We analyze the mode of JH hash function, another final round candidate of SHA3 competition. JH uses a novel domain extension algorithm, somewhat reminiscent of a sponge construction, to build a hash function out of a single, fixed $2n$-bit permutation using chopped-MD domain extension. In this thesis, we present the first detailed security analysis of JH mode of operation. We also consider a modified mode of operation of JH where the chopping is done on the other bits. We prove that under the assumption that the fixed permutation of JH is a random permutation, JH mode of operation with specific length padding and the modified JH mode of operation *without* length padding is indifferentiable from a Random Oracle. We also show that the padding is *essential* for JH domain extension as one can construct a distinguisher that can distinguish (in the framework of indifferentiability) the output of a Random Oracle from JH mode of operation without padding with n-bit output using only *constant* number of queries.

Secure Instantiation of Public Key Constructions

In the second half of the thesis, we concentrate on black-box reduction of popular encryption and signature schemes when the underlying hash function, given to the adversary, is not modeled as a Random Oracle. First, we consider the padding based signature schemes where the random string is recoverable. One of popular example of such padding based signature schemes is PSS which has been standardized in numerous standards like PKCS# 1.

Our main result on the padding based signature is a general blackbox separation result. Extending the work of Dodis et. al. on Full Domain Hash, we show that there is no instantiation of "randomization recoverable" padding based probabilistic signature scheme that can be reduced to any security property of an ideal trapdoor permutation or a lossy trapdoor permutation in *black-box* manner. As an ideal trapdoor permutation satisfies almost all reasonable security notions, our result covers many of the standard security notions, like inverting trapdoor permutation on a random point (one-way), finding some bits of preimage of a random point (partial domain one-wayness), finding correlated inputs etc. Our result is perfectly valid even if the hash functions used in the padding are arbitrarily correlated with the trapdoor permutation. Moreover our result holds even in the scenario when the adversary can invert some point of his choice (with some restrictions) for a fixed bounded number of times. Moreover we show that even the weakest security criterion, namely unforgeability under *no message attack* of popular Probabilistic Signature Scheme (PSS) cannot be black-box reduced to the one-wayness of the trapdoor permutation as long as the randomness space is "super-logarithmic" in the security parameter. This result shows that use of padding does not help to prove security in the standard model.

Finally, we concentrate on instantiation of padding based encryption schemes, specifically OAEP, by the seed incompressible functions. OAEP is one of the most popular and widely deployed asymmetric encryption schemes. It was designed by Bellare and Rogaway as a padding scheme for encryption using a trapdoor permutation such as RSA. OAEP is standardized in RSA's PKCS #1 v2.1 and is part of the ANSI X9.44, IEEE P1363, ISO 18033-2 and SET standards. However security of OAEP against chosen ciphertext attack has been proved *only* in Random Oracle Model. In fact, similar to our result for signatures, Kiltz and Pietrzak (in Eurocrypt'09) has proved that, CCA security of padding based encryption scheme in the standard model cannot be reduced to any security property of an ideal trapdoor permutation using black-box techniques. In this thesis we analyze CCA security of OAEP in the seed incompressible function model where the adversary can have some short string (compared to the key-size of the hash function) as an advice. We show that, if the length of the advice is allowed to be twice more than the square of the size of the padded message, then CCA security of OAEP in the standard model cannot reduced to any security property of an ideal trapdoor permutation using black-box techniques. We also show that when the length of the randomness is small compared to the security parameter and the advice string is allowed to be in the cubic order the size of the random string, same result holds. We also extend our results for the case of lossy trapdoor permutations. Our technique is quite generic and can be applied to any padding based encryption scheme. *To the best of our knowledge, our result is the first impossibility result of any encryption scheme in the seed incompressibility model.*

1.4 Organization of the Thesis

The thesis is divided in two parts. In the first part, we consider the constructions from Symmetric key Cryptography and analyze their security in the indifferentiability framework of Maurer. Reductions and impossibility results of public key cryptosystems are considered in the second part. Overall, the thesis is organized in the following way.

Part I: Indifferentiability of Modes of Operation

Chapter 2: In this chapter, we discuss the definitions of different security properties of the mode of operation and survey the existing results on Indifferentiability of hash functions.

Chapter 3: We find sufficient condition of indifferentiability for all domain extension algorithms in GDE. We apply these conditions to existing hash functions (e.g. Blake, Skein, MD6) and achieve optimal bounds. This chapter is based on a paper published in Indocrypt'09.

Chapter 4: In this chapter, we apply the techniques described in Chapter 2 to JH hash function, a SHA3 final round candidate and analyze indifferentiability of corresponding mode of operation. We also consider a variant of JH mode of operation and achieve better indifferentiability bound than the actual JH mode. Results of this chapter is based on our publication in FSE'10.

Part II

Chapter 5: In this section we review the security notions of public key encryption and signatures followed by the definitions and the techniques used for our results. We also present the existing results related to the thesis.

Chapter 6: In this chapter, we prove the impossibility of proving security of Padding based Signature Schemes in the standard model, where the underlying hash function is unkeyed, fixed. We present impossibility results for One-way Trapdoor Permutations, Ideal Trapdoor Permutations (a generalization of trapdoor permutations by Kiltz and Pietrzak) as well as Lossy Trapdoor Permutations. This chapter is based on our paper in PKC'11.

Chapter 7: This chapter is on chosen ciphertext security of *Optimal Asymmetric Encryption Padding* (OAEP) in Seed Incompressibility model. We show impossibility of black-box reduction of OAEP from ideal trapdoor permutation and lossy trapdoor permutation even when the adversary can have some information about the hash function as an oracle.

Chapter 8: This chapter concludes the thesis. We present a summary of the results presented in the thesis. Finally we end the chapter and the thesis with a discussion on the open problems in the area of black-box reduction in Cryptography.

1.5 Notation

Throughout the thesis, we follow the following notations. 1^n denotes the string of n many 1s. If S is a set $|S|$ denotes the cardinality of the set. If x is a string, $|x|$ denotes the length of the string. $x \leftarrow_{\text{R}} S$ denotes that x is chosen uniformly at random from the set S. We use $negl(n)$ to denote any function $\gamma : \mathbb{N} \to [0, 1]$ where for any constant $c > 0$ there exists n_0 such that for all $n > n_0$; $\gamma(n) < 1/n^c$. We call a function $f(n)$ to be super-polynomial if for any constant $c > 0$, there exists n_0 such that for all $n > n_0$, $f(n) > n^c$.

Part I

Blackbox Reduction of Modes of Operations

Chapter 2

Blackbox Reduction of Cryptographic Hash Functions

2.1 Introduction

Designing secure hash function is a primary objective of symmetric key cryptography. A cryptographic hash function is an efficiently computable function $H : \{0,1\}^* \rightarrow \{0,1\}^n$ for some fixed integer n. The following security properties are usually expected from a cryptographic hash function.

- **Preimage Resistance** Given a randomly selected element $z \in \{0,1\}^n$, it should be "infeasible" to find a M such that $H(M) = z$. Note that, any element in $\{0,1\}^n$ might have more than one preimages under H. The security requirement ensures that an adversary will not be able to find *any* preimage of a hash digest.

- **Second Preimage Resistance** Given a random message $M \in \{0,1\}^*$, it should be "difficult" to find another message M' such that $H(M) = H(M')$.

- **Collision resistance** It should be "infeasible" to find two *distinct* messages $M, M' \in \{0,1\}^*$ such that $H(M) = H(M')$.

The terms "infeasible" or "difficult" are used for adversaries with bounded (polynomial) running time or query complexity. Many more requirements like target collision resistance, multicollision resistance etc. are known. However, the above three conditions are most common requirements from a cryptographic hash function.

2.2 Mode of Operation of a Hash Function

Popular methods to build a hash function involve two steps. First, one designs a compression function $f : \{0,1\}^m \to \{0,1\}^n$ where $m > n$. Then a domain extension algorithm $\mathcal{C}$ that utilizes f as a black box is applied to implement the hash function $C^f : \{0,1\}^* \to \{0,1\}^n$. This is also known as design or mode of the hash function.

Essentially any mode of operation is characterized by two properties

- **Padding Rule** The input to the underlying compression function is of fixed length (m). Usually, the input message is divided into disjoint fixed size blocks and one block is used in each iteration. To hash the messages whose lengths are not multiple of block size, some padding rule is applied, so that the length of the padded message becomes a multiple of block size. A simple padding rule can be zero padding, that is adding sufficient number of zero bits so that the padded message becomes a multiple of block size. Sometimes, padding rules are also used to avoid trivial attacks.

- **Input functions** The input function defines the input to the next query to compression function f. For iterated hash functions input function may take a message block along with outputs of previous f queries and the initialization vectors as input.

2.2.1 Examples of Domain Extension of Hash Function

Merkle-Damgård domain extension

The most popular domain extension algorithm is Merkle-Damgård domain extension technique. The (padded) message is divided in $m - n$ bit blocks. The first message block along with some fixed initialization vector is queried to the compression function f. At each subsequent step, one block is appended with the output of the previous query and queried to f. The output of the last query is the output of the domain extension algorithm.

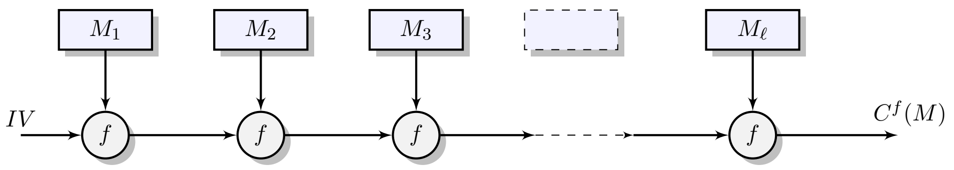

Figure 2.1: The Merkle-Damgård domain extension technique

Chopped Merkle-Damgård domain extension

In case of Chopped Merkle-Damgård (Chop-MD) mode, some bits from the output of Merkle-Damgård domain extension are chopped and rest of the bits are given as output. In Chapter 4 we shall consider a chopped MD domain extension technique.

Envelope Merkle-Damgård domain extension

In envelope Merkle-Damgård mode, after applying the usual Merkle-Damgård mode, the chaining value is used to query an independent (to the compression function) function and the corresponding response is published as the digest.

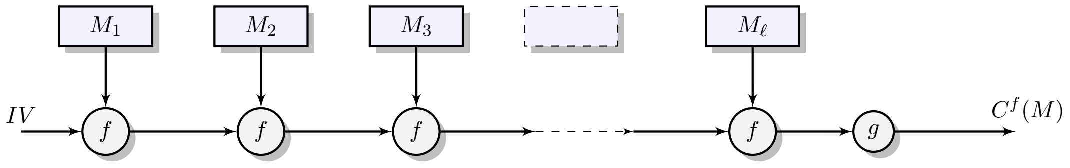

Figure 2.2: The Envelope Merkle-Damgård domain extension technique

Tree mode of operation

In case of tree mode, the domain extension algorithm queries the compression function like a k-ary tree. The messages are divided in m bit blocks and queried to the leaf level. At every other level, input to a f query is the concatenation of the outputs of its child nodes. Finally, the output of the root node is the output of the mode.

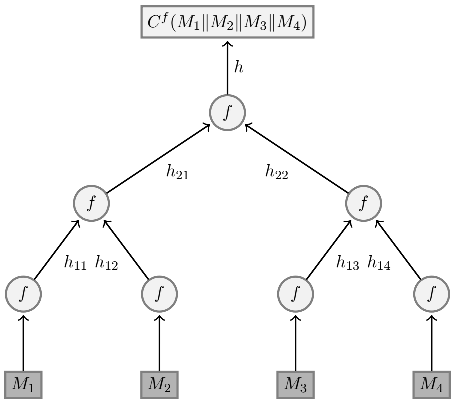

Figure 2.3: The Tree mode of operation

In Chapter 2, we consider the tree mode of operation with a specific padding rule.

Sponge domain extension

Sponge domain extension, introduced by Bertoni et. al. is a permutation based hash function design. Let $\pi^{r+c} \to \pi^{r+c}$ be a fixed unkeyed permutation. The sponge mode works in two stages. In the first stage, known as the absorption stage, the message blocks of length r bits, are xored with the first r bits of the chain and passed through the permutation one by one. In the second stage (squeezing), digests are formed by taking some arbitrary, fixed part of the chaining value at each iteration. This allows the sponge design to output a digest of arbitrary size without changing the size of the permutation. r is called the block-length and c is called the capacity of the domain extension. Note that, $|c|$ need not be same as $|r|$.

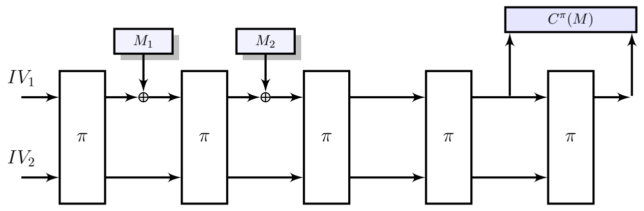

Figure 2.4: The Sponge mode of operation

2.2.1.1 Security Properties of Mode of Operation

Intuitively a good domain extension algorithm should maintain the security property of the underlying compression function. For example, if the compression function f is collision resistant (i.e. it is infeasible to find $x_1, x_2 \in \{0,1\}^m$ with $f(x_1) = f(x_2)$), C^f should be collision resistant as well.

Definition 2.2.1. Collision Resistance of Domain Extension A domain extension algorithm C, based on a compression function f is said to maintain the collision resistance, if there exists an efficient reduction $\mathcal{R}$ such that for every adversary $\mathcal{A}$ with

$$Prob[(M, M') \leftarrow \mathcal{A}^{C^f}(.); C^f(M) = C^f(M')] \geq \delta$$

there exists a non-negligible ϵ

$$Prob[(M, M') \leftarrow \mathcal{R}^{\mathcal{A}^{C^f}, f}; f(M) = f(M')] \geq \epsilon\delta.$$

Preimage Resistance and Second Preimage Resistance of a domain extension algorithm can be defined in similar fashion. However, in modern days, domain extension algorithm should also maintain the properties of Random Functions.

2.3 Random Functions and Random Permutations

Ideally, a cryptographic hash function should be secure against any kind of attack. There should not be any particular relations between the digests of any number of messages. Moreover, the digests should be, in some sense, "independent" of the messages. The hash function should not have (in functional sense) any structure.

This requirement essentially demands the hash function to behave like a *Random Function*. Intuitively a random function (oracle) is a function $f : X \to Y$ chosen uniformly at random from the set of all functions from X to Y. The following is an equivalent definition of a random function.

Definition 2.3.1. Random Oracle [81] $f : X \to Y$ is said to be a *random oracle* if for each $x \in X$ the value of $f(x)$ is chosen uniformly at random from Y. More precisely, for $x \notin \{x_1, \ldots, x_q\}$ and $y, y_1, \cdots, y_q \in Y$ we have

$$\Pr[f(x) = y \mid f(x_1) = y_1, f(x_2) = y_2, \cdots, f(x_q) = y_q] = \frac{1}{|Y|}$$

A random permutation is similar to random oracle except that it is a permutation. So similarly one can view a random permutation $\pi : X \to X$ as a permutation chosen uniformly at random from the set of all permutations over X. Similar to random oracles, we have the following equivalent definition of random permutation.

Definition 2.3.2. Random Permutation[81] $\pi : X \to X$ is said to be a *random permutation* if for each $x \in X$ we have,

$$\Pr[\pi(x) = y \mid \pi(x_1) = y_1, \pi(x_2) = y_2, \ldots, \pi(x_q) = y_q] = \frac{1}{|X| - q}$$

where $|X|$ is finite and $x \notin \{x_1, \ldots, x_q\}$, $y_1, \ldots, y_q \in X$ and $y \in X \setminus \{y_1, \ldots, y_q\}$

2.3.1 Adversary in the (Random) Oracle Model

In case of blackbox reduction of a hash function C^f, from a (random) compression function f, the adversary is modeled as an oracle algorithm $\mathcal{A}^f$ with only oracle access to f. The adversary can query f adaptively and based only on the query-response the adversary produces some output. Thus $\mathcal{A}$ can adaptively choose $x_1, \cdots, x_q$ and get $y_1, \cdots, y_q$ where $y_i = f(x_i)$ for all $i = 1, \cdots, q$. We call this list of query response $\{(x_1, y_1), \cdots, (x_q, y_q)\}$ as the *view* of the adversary. The output of the adversary should only depend on the view and its internal randomness. For example, for an adversary producing collision (M, M') for the hash function, the relation $C^f(M) = C^f(M')$ should be computable from the view. The complexity of the adversary is measured by the size of its view.

In case of permutation based hash function, the adversary will have access to two oracles; the permutation π and its inverse π^{-1}. Similar to the previous case, if an adversary outputs some message M, it must have made all required π and π^{-1} queries to compute $C^{\pi}(M)$. Note that, this requirement does not handicap the adversary in any way. Indeed, for any adversary $\mathcal{A}$ not satisfying this relation there exists an equivalent adversary $\mathcal{A}'$ which maintains the condition and has equivalent success probability.

The security of a hash function can be measured by the minimum complexity of an adversary with significant success probability.

2.3.2 Hash Functions as Random Oracles

Random Oracle paradigm, introduced by Bellare and Rogaway [12], is a very popular platform for proving security of cryptographic protocol. In this model all the participating parties, including the adversary, is given access to a truly random function R. Unfortunately, it is impossible to realize a truly random function in practice. So while implementing the protocol the most natural choice is to instantiate R by an *ideal* hash function H. The formal proofs in Random Oracle model indicate that there is no structural flaw in the designed protocol. But how can we make sure, that replacing the random function R with a *good* hash function H will not make the protocol insecure? In fact recent results [30, 91] show that theoretically it is possible to construct some pathological protocols that are secure in random oracle model but completely insecure in standard model. Fortunately those separation results do not imply an immediate serious threat to any widely used cryptosystem, proven to be secure in random oracle model. So one can hope that any attack, which fails when a protocol is instantiated with R but succeeds when the protocol is instantiated with H, will use some structural flaw of H itself. So the above question boils down to the following. *How can we guarantee the structural robustness of a hash function H?*

2.4 Indifferentiability

The notion of indifferentiability was introduced by Maurer et. al. in [86] as a generalization of indistinguishability. Loosely speaking, if an ideal primitive $\mathcal{G}$ is indifferentiable from a construction C based on another ideal primitive $\mathcal{F}$, then $\mathcal{G}$ can be safely replaced by $C^{\mathcal{F}}$ in any cryptographic construction. In other terms if a cryptographic construction is secure in $\mathcal{G}$ model then it is secure in $\mathcal{F}$ model.

Definition 2.4.1. Indifferentiability [86]
A Turing machine C with oracle access to an ideal primitive $\mathcal{F}$ is said to be $(t, q_C, q_{\mathcal{F}}, \varepsilon)$ indifferentiable from an ideal primitive $\mathcal{G}$ if there exists a simulator S with an oracle access to $\mathcal{G}$ and running time at most t, such that for any distinguisher D,

$$\mathbf{Adv}^{indiff}_{(C^{\mathcal{F}},\mathcal{F}),(\mathcal{G},S^{\mathcal{G}})} = |\Pr[D^{C^{\mathcal{F}},\mathcal{F}} = 1] - \Pr[D^{\mathcal{G},S^{\mathcal{G}}} = 1]| < \varepsilon.$$

The distinguisher makes at most q_C queries to C or $\mathcal{G}$ and at most $q_{\mathcal{F}}$ queries to $\mathcal{F}$ or S. Similarly, $C^{\mathcal{F}}$ is said to be (computationally) indifferentiable from $\mathcal{G}$ if running time of D is bounded by some polynomial in the security parameter k and ε is a negligible function of k.

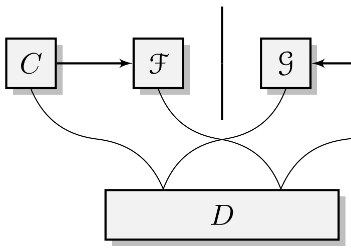

Figure 2.5: The indifferentiability notion

We stress that in the above definition $\mathcal{G}$ and $\mathcal{F}$ can be two completely different primitives. As shown in Fig 2.5 the role of the simulator is to not only simulate the behavior of $\mathcal{F}$ but also remain consistent with the behavior of $\mathcal{G}$. Note that, the simulator does not know the queries made directly to $\mathcal{G}$, although it can query $\mathcal{G}$ whenever it needs.

Recall that, a cryptosystem can be modeled as an interactive Turing Machine with an interface to an adversary and to a public oracle. The cryptosystem is run by an environment (modeled as a code based game) which also runs the adversary and produces some output. The following theorem is a concrete version of composition theorem proved in [86].

Theorem 2.4.2. *[99] Let G be a single stage game expecting access to a functionality and a single adversarial procedure. Let $\mathcal{F}$ and $\mathcal{G}$ be two functionalities and compatible honest interfaces. Let $\mathcal{A}$ be an adversary with one oracle. Let S be a simulator that exports the adversarial interface of $\mathcal{F}$. Then there exist adversary $\mathcal{B}$ and distinguisher D such that for all values of y*

$$Prob[G^{\mathcal{G},\mathcal{A}} = y] \leq Prob[G^{\mathcal{F},\mathcal{A}} = y] + \textbf{\textit{Adv}}^{indiff}_{\mathcal{F},\mathcal{G},S}$$

Moreover, $t_{\mathcal{B}} \leq t_{\mathcal{A}} + q_{\mathcal{A}} \cdot t_S$, $q_{\mathcal{B}} \leq q_{\mathcal{A}} \cdot q_S$, $t_D \leq t_G + q_{G,1} \cdot t_{\mathcal{A}}$ and $q_D \leq q_{G,0} + q_{G,1} \cdot q_{\mathcal{A}}$, where $t_{\mathcal{A}}, t_{\mathcal{B}}, t_D$ are the running time of $\mathcal{A}$,$\mathcal{B}$,D, $q_{\mathcal{A}}, q_{\mathcal{B}}$ are the maximum number of queries made by $\mathcal{A}$ and $\mathcal{B}$ in a single execution, $q_{G,0}, q_{G,1}$ are the maximum number of queries made by G to the honest interface and the adversarial procedure.

In simple terms, the above theorem says that, if there exists a construction C , based on ideal primitive $\mathcal{F}$, implementing the functionality of another ideal primitive $\mathcal{G}$ and $C^{\mathcal{F}}$ is indifferentiable to $\mathcal{G}$, then one can replace $\mathcal{G}$ by $C^{\mathcal{F}}$ in any single stage cryptosystem and the cryptosystem will be as secure in $\mathcal{F}$ model as in $\mathcal{G}$ model.

Indifferentiability of Hash Functions: Based on the framework of Maurer et. al., Coron, Dodis, Malinaud and Puniya formalized the notion of Indifferentiability of hash functions. The ideal primitive

$\mathcal{G}$ is replaced by a random oracle $\mathcal{R}$. The interface $\mathcal{F}$ represents a fixed input length random oracle (FIL-RO) or a random permutation depending on the mode of operation. In the following definition, we restate the notion of indifferentiability of a hash function.

Definition 2.4.3. Indifferentiability of Hash Function[36]
A domain extension C with oracle access to an ideal primitive $\mathcal{F}$ is said to be $(t, q_C, q_{\mathcal{F}}, \varepsilon)$ indifferentiable from a Random Oracle $\mathcal{R}$ if there exists a simulator S with an oracle access to $\mathcal{R}$ and running time at most t, such that for any distinguisher D,

$$\mathbf{Adv}^{indiff}_{(C^{\mathcal{F}},\mathcal{F}),(\mathcal{R},S^{\mathcal{R}})} = |\Pr[D^{C^{\mathcal{F}},\mathcal{F}} = 1] - \Pr[D^{\mathcal{R},S^{\mathcal{R}}} = 1]| < \varepsilon.$$

The distinguisher makes at most q_C queries to C or $\mathcal{R}$ and at most $q_{\mathcal{F}}$ queries to $\mathcal{F}$ or S. Similarly, $C^{\mathcal{F}}$ is said to be computationally indifferentiable from $\mathcal{R}$ if running time of D is bounded by some polynomial in the security parameter k and ε is a negligible function of k.

2.5 Existing Works on Indifferentiability

Since the formalization of Coron et. al. numerous hash designs have been proved indifferentiable from a random oracle assuming the basic building block (the compression function or the permutation) is ideal. In this section, we review the existing results on the indifferentiability of the hash functions.

2.5.1 Indifferentiability of Merkle-Damgård with prefix free padding

A prefix free coding over an alphabet $\{0,1\}$ is an injective map $\text{PAD} : \{0,1\}^* \to \{0,1\}^*$, such that for all distinct x, y, $\text{PAD}(x)$ is not a prefix of $\text{PAD}(y)$. Moreover, there must be a decoding algorithm DEPAD such that for all $x \in \{0,1\}^*$, $\text{DEPAD}(\text{PAD}(x)) = x$. Simple example of prefix free encoding is to divide the message into equal length blocks. Prepend the length of the string as a first block with last block padded with 10^r. In [36], Coron et. al. proved the following theorem

Theorem 2.5.1. *[36] The prefix free Merkle-Damgård domain extension based on a n-bit FIL-RO is indifferentiable from a Random Oracle for $t = \ell\mathcal{O}(q^2)$ and $\epsilon = \mathcal{O}(\frac{q^2\ell^2}{2^n})$ where q is the total number of queries made by the distinguisher, ℓ is the maximum number of blocks in a single query.*

2.5.2 Indifferentiability of Merkle-Damgård tree mode

The Merkle-Damgård domain extension can also be used in a tree mode. In [44], Dodis et al. proved the following theroem

Theorem 2.5.2. *[44] The Merkle-Damgård tree mode of operation based on a n-bit FIL-RO is indifferentiable from a Random Oracle for $t = \ell\mathcal{O}(q^2)$ and $\epsilon = \mathcal{O}(\frac{q^2\ell^2}{2^n})$ where q is the total number of queries made by the distinguisher, ℓ is the maximum number of blocks in a single query.*

In Theorem 3.4.2, we improve this bound to $\epsilon = \mathcal{O}(\frac{q^2 \log \ell}{2^n})$.

2.5.3 Indifferentiability of Chop-MD domain extension

Recall that, in case of Chop-MD domain extension, some bits (say s) of the output of the final query to the compression function is chopped and the rest of the bits are used to form the hash digest. Coron et. al. proved the following results for chopped Merkle-Damgård domain extension without prefix free padding

Theorem 2.5.3. *[36] The chopped Merkle-Damgård domain extension based on a n-bit FIL-RO and s-bit chopping is indifferentiable from a Random Oracle for $t = \ell\mathcal{O}(q^2)$ and $\epsilon = \mathcal{O}(\frac{q^2\ell^2}{2^s})$ where q is the total number of queries made by the distinguisher, ℓ is the maximum number of blocks in a single query.*

In [32], Chang and Nandi improved the the bound in the following lemma

Theorem 2.5.4. *[32] The chopped Merkle-Damgård domain extension based on a n-bit FIL-RO and s-bit chopping is indifferentiable from a Random Oracle for $t = \ell\mathcal{O}(q^2)$ and $\epsilon = \mathcal{O}(\frac{q(n-s)}{2^s} + \frac{q}{2^{n-s}} \frac{q^2\ell^2}{2^n})$ where q is the total number of queries made by the distinguisher, ℓ is the maximum number of blocks in a single query.*

2.5.4 Indifferentiability of Envelope MD domain extension

In case of Envelope MD the output of the final compression function f is queried to an independent function g and the corresponding output is given as digest. Bellare and Ristenpart, proved the indifferentiability of Envelope MD construction when both f and g are modeled as independent FIL-RO.

Theorem 2.5.5. *[10] The Envelope Merkle-Damgård domain extension based on two independent n-bit FIL-RO is indifferentiable from a Random Oracle for $t = \ell\mathcal{O}(q^2)$ and $\epsilon = \mathcal{O}(\frac{q^2\ell^2}{2^n})$ where q is the total number of queries made by the distinguisher, ℓ is the maximum number of blocks in a single query.*

2.5.5 Indifferentiability of Sponge Domain Extension

Indifferentiability of Sponge construction was proven by Bertoni et. al. [15]. This was the first Indifferentiability result based on fixed random permutation.

Theorem 2.5.6. *[15] The Sponge domain extension based on $r + c$-bit random permutation is indifferentiable from a Random Oracle for $t = \ell\mathcal{O}(q^2)$ and $\epsilon = \mathcal{O}(\frac{q^2\ell^2}{2^c})$ where q is the total number of queries made by the distinguisher, ℓ is the maximum number of blocks in a single query, r is the size of the processed message block in each iteration.*

Chapter 3

Towards a Unified Proof Technique for Indifferentiability

In this chapter, we present a unified technique to prove indifferentiability of a wide class of hash designs. Although many known hash function constructions have been shown to be indifferentiable from an RO, the proof of these results are usually complicated (many times, due to numerous game hopings and hybrid arguments). Also, they require different simulators for each individual hash design. There are no known sufficient conditions for hash functions to be indifferentiable from an RO. From a different perspective, the existing security bounds for different constructions are not always optimal[1]. The results of [45, 85] do not directly imply to improve the indifferentiability bounds for general iterated hash functions based on a single random oracle. The methods of [44] do not give us any optimal bound either. So a natural question to ask is: *Can we characterize the minimal conditions of a cryptographic hash function to be indifferentiable from a Random Oracle and achieve optimal bound?*

Our Result: In this chapter, we present a unified technique of proving indifferentiable security for a major class of iterated hash functions, called Generalized Domain Extensions. We extend the technique of [85] to the indifferentiability framework. We identify a set of events (called BAD events) and show that any distinguisher, even with unbounded computational power, has to provoke the BAD events in order to distinguish the hash function C from a random function R. Moreover we prove that, to argue indifferentiability of a construction C^f, one has only to show that the probability that any distinguisher invokes those BAD events, while interacting with the pair (C^f, f), is negligible. **We avoid the cumbersome process of defining simulator for each construction separately by providing a unified simulator for a wide range of constructions. To prove indifferentiability one simply needs to compute the probability of provoking the BAD events when interacting with (C^f, f).**

[1] In fact, to the best of our knowledge, none of the known bounds except the one of sponge construction, was proven to be tight.

In the second part of this chapter, we apply our technique to some popular domain extension algorithms to provide optimal indifferentiability bounds. In particular, we consider Merkle-Damgård with HAIFA and tree mode with specific counter padding. Many candidates of *SHA3* competition actually use these two modes of operations. So, our result can also be viewed as an **optimal** indifferentiability guarantee of these candidates. We briefly describe our results below:

1. **MD with counter** or HAIFA: Let C^f be MD with counter where the last block counter is zero (all other counters are non-zero). Many SHA3 candidates such as BLAKE, LANE, SHAvite-3 etc are in this category. In Theorem 3.4.1 and Theorem 3.5.2, we show that the (tight) indifferentiable bound for C is $\Theta(\sigma q/2^n)$ where q is the number of queries, n is the size of the hash output and σ is total number of message blocks in all the queries. The so far best known bound for HAIFA mode is $\sigma^2/2^n$ [36].

2. **Tree-mode with counter**: Tree mode with counter (e.g. the mode used in MD6) is known to be indifferentiable secure with upper bound $q^2\ell^2/2^n$ [44]. In Theorem 3.4.2 and Theorem 3.5.3, we are provide an optimal indifferentiable bound $\Theta(q^2 \log \ell/2^n)$.

3.1 Generalized Domain Extension (GDE)

Intuitively, any practical domain extension technique applies the underlying compression function f in a sequence, where inputs of f are determined by previous outputs and the message $M \in \{0,1\}^*$ (for parallel constructions, inputs only depend on the message). Finally the output $C^f(M)$ is a function of all the previous intermediate outputs and the message M. The *Generalized Domain Extension* (GDE) are the domain extension techniques where u_ℓ is the input to final invocation of f and $C^f(M) = f(u_\ell)$. A domain extension algorithm from the class GDE is completely characterized by the following two functions:

1. **Length function**: $\ell : \{0,1\}^* \to \mathbb{N}$ is called *length function*, which actually measures the number of invocations of f. More precisely, given a message $M \in \{0,1\}^*$, $\ell = \ell(M)$ denotes the number of times f is applied while computing $C^f(M)$.

2. **Input function**: For each $j \in \mathbb{N}$, $U_j : \{0,1\}^* \times (\{0,1\}^n)^j \to \{0,1\}^{m'}$, called j^{th} *input function*. It computes the input of j^{th} invocation of f. This is computed from the message M and all $(j-1)$ previous outputs of f. In other words, $U_j(M, v_0, v_1, \cdots, v_{j-1})$ is the input of j^{th} invocation of f while computing $C^f(M)$, where $v_1, \cdots, v_{j-1}$ denote the first $(j-1)$ outputs of f and v_0 is a constant depending on the construction. The input function usually depends on message block, instead of whole message and hence we may not need to wait to get the complete message to start invoking f.

The above functions are independent of the underlying function f. Note that the padding rule of a domain extension is implicitly defined by the input functions defined above. At first sight, it may seem that GDE does not capture the constructions with independent post processor. But we argue that, when the underlying primitive is modeled like a random oracle, then queries to the post processor can be viewed as queries to same oracle (as in the intermediate queries) but with different padding. Namely in case of NMAC like constructions, we can consider a GDE construction where the inputs to the intermediate queries are padded with 1 and the final query is padded with 0. Similarly, one can incorporate domain extensions which use more than one random oracle.

Definition 3.1.1. *(GDE*: **Generalized Domain Extension***)*
Let $\mathcal{S} = (\ell, \langle U_j \rangle_{j \geq 1})$ be tuple of deterministic functions as stated above. For any function $f : \{0,1\}^{m'} \to \{0,1\}^n$ and a message M, $\mathit{GDE}^f_{\mathcal{S}}(M)$ is defined to be v_ℓ, where $\ell = \ell(M)$ and for $1 \leq j \leq \ell$,

$$v_j = f\big(U_j(M, v_0, v_1, \cdots, v_{j-1})\big).$$

The $u_j = U_j(M, v_0, v_1, \cdots, v_{j-1})$ is called the j^{th} intermediate input for the message M and the function f, $1 \leq j \leq \ell$. Similarly, $v_j = f(u_j)$ is called j^{th} intermediate output, $1 \leq j \leq \ell - 1$. The last intermediate input u_ℓ is also called final (intermediate) input. The tuple of functions $\mathcal{S}$ completely characterizes the domain extension and is called the **structure** of the domain extension $\mathit{GDE}_{\mathcal{S}}$.

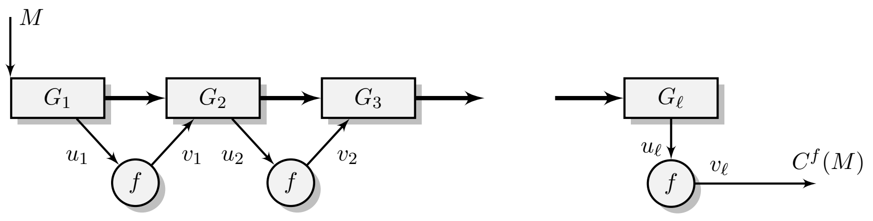

Figure 3.1: The Generalized Domain Extension Circuit

Note that we can safely assign $v_0 = IV$, the Initialization Vector, used in many domain extensions. In Fig 3.1 we describe the concept of GDE. Each G_i is an algorithm which computes the i^{th} intermediate input u_i, using the input-function U_i defined above. Note that, we have allowed U_i to take the entire message as the input. However, in practice, it may depend only on few or one message block. The wires between G_i and G_{i+1} is thick. In fact it contains all the previous input, output and the state information. In this chapter we describe sufficient conditions to make a Generalized Domain Extension technique indifferentiable from a Random Oracle (RO). In the next section we show a hybrid technique to characterize the conditions and prove its correctness.

3.2 Indifferentiability of GDE

In this section we discuss the sufficient condition for a domain extension algorithm C of the class GDE to be indifferentiable from a random oracle $\mathcal{R}$. Let $C \in$ GDE be a domain extension algorithm based on a fixed input length random oracle f. Recall that to prove the indifferentiability, for any distinguisher D running in time bounded by some polynomial of the security parameter κ, we need to define a simulator S such that

$$|\Pr[D^{C^f,f} = 1] - \Pr[D^{\mathcal{R},S^{\mathcal{R}}} = 1]| < \varepsilon(\kappa).$$

Here $\varepsilon(\kappa)$ is a negligible function and the probabilities are taken over random coin tosses of D and randomness of f and R. Let *left query* denote the queries to R/C^f and *right query* denote the queries to S^R/f. The simulator keeps a list L, initialized to empty. If u_i is the i^{th} query to the simulator and the response of the simulator was v_i then the i^{th} entry of L is the tuple (i, u_i, v_i).

Definition 3.2.1. Let $C \in$ GDE. We say that $C^f(M)$ for a message M is computable from a list $L = \{(1, u_1, v_1), \cdots, (k, u_k, v_k)\}$ if there are $\ell = \ell(M)$ tuples (i_1, u_{i_1}, v_{i_1}),$\cdots$, $(i_\ell, u_{i_\ell}, v_{i_\ell}) \in L$ such that for all $t \in \{1, 2, \cdots, \ell\}$,

$$u_{i_t} = U_t(M, v_0, v_{i_1}, \cdots, v_{i_{t-1}}).$$

Intuitively for any simulator to work, C must have the following property:

Message Reconstruction: There should be an efficient algorithm $\mathcal{P}$[1] such that given a set $L = \{(1, u_1, v_1), \cdots, (k, u_k, v_k)\}$, input-output of k many f queries and an input $u \in \{0,1\}^{m'}$ (in the domain of f); $\mathcal{P}(L, u)$ outputs M if $C^f(M)$ is computable from $L \cup \{(k+1, u, v)\}$ for all $v \in \{0,1\}^n$ where $u_\ell = u$ (as in Definition 3.2.1). If no such M exists $\mathcal{P}$ outputs $\perp$. If there are more than one such M, we assume $\mathcal{P}$ outputs any one of them.[2]

We argue that this is a very general property and is satisfied by all known secure domain extensions. In fact, the Message reconstruction algorithm $\mathcal{P}$ defined above is similar to the extractor of Preimage Awareness (PrA) of [45]. This is very natural as the notion of PrA is much relaxed notion than the notion of PRO and every PRO is essentially PrA [45]. However existence of such an algorithm does not guarantee indifferentiability from a Random Oracle. For example, the traditional Merkle-Damgård construction is PrA but not PRO. In fact, the method of [45] is only applicable to prove indifferentiability when the final query is made to an independent post processor. On the other hand, our contribution in this chapter is to show a set of sufficient conditions along with the existence of extractor for a domain extension of the class GDE (where the final query can be made to that same function) to be a PRO.

[1]Note that the exact description of $\mathcal{P}$ depends on specific implementation.

[2]For example, $\mathcal{P}$ can choose a message randomly among all such messages. However, it will actually invoke BAD event.

Our simulator works as follows. Suppose the k^{th} query to the simulator is u. Then

- If $(i, u, v) \in L$ for some $i < k$ and some $v \in \{0,1\}^n$, then $L = L \cup \{(k, u, v\}$ and return v.
- If $\mathcal{P}(L, u) = M$
  - $L = L \cup \{(k, u, R(M))\}$
  - return $R(M)$
- If $\mathcal{P}(L, u) = \perp$
  - Sample $h \in_R \{0,1\}^n$
  - $L = L \cup \{(k, u, h)\}$
  - return h

Without loss of generality, we can assume adversary maintains two lists L_{left} and L_{right} to keep the query-responses made to R/C^f and S^R/f respectively.

3.2.1 Security Games

To prove the indifferentiability of GDE we shall use hybrid technique. We start with the scenario when the distinguisher D is interacting with C^f, f.

```
A left query C(M)
    1. $v_0 = \lambda$.
    2. $\ell = \ell(M)$.
    3. for $i = 1$ to $\ell$
        (a) $u_i = U_i(M, v_0, v_1, \cdots, v_{i-1})$.
        (b) $v_i = COM\_RO(u_i)$.
    4. return $v_\ell$.

A right query S(u)
    1. return $COM\_RO(u)$.

COM_RO($u$)
    1. return $f(u)$.
```

Figure 3.2: Procedures of Game 0

Game 0: In this game the distinguisher is given access to an oracle S for the right queries. Additionally, both C and S are given access to another oracle COM_RO which can make f queries. Note that C or S do not have direct access to f. S on an input (u), queries $COM\_RO(u)$. COM_RO on input u

returns $f(u)$. Formally, Game 0 can be viewed as Fig 3.2. Since the view of the distinguisher remains unchanged in this game we have

$$Pr[D^{C^f, f} = 1] = Pr[G_0 = 1]$$

where G_0 is the event when the distinguisher outputs 1 in Game 0.

Game 1 Now we change the description of the subroutine COM_RO and gives it an access to random oracle R as well. In this game COM_RO takes a 3-tuple (u, M, tag) as input where $u \in \{0,1\}^{m'}, M \in \{0,1\}^m$ and $tag \in \{0,1\}$. COM_RO returns $f(u)$ when $tag = 0$ and returns $R(M)$ otherwise. We also change the procedure to handle left and right queries. In this game, the algorithm S maintains a list L containing the query number, input, output of previous right queries. While processing a left query M, the algorithm queries COM_RO with $tag = 1$ when querying with u_ℓ and makes $tag = 0$ for all other queries. Informally speaking, for a left query M, the algorithm C behaves almost similarly as game 0, except it returns $R(M)$ as the response. Similarly when a right query is a trivially derived from L and some message M, the algorithm sets $tag = 1$ before querying COM_RO and sets $tag = 0$ otherwise. Formally $Game1$ can be viewed as Figure 3.3. In the figure, the variable $index$ represents the number of distinct queries made to S, so far; i. e. $index$ is the size of the list L. Initially index is set to 0. λ represents the empty string.

Definition 3.2.2. Trivial Query
A right query u is said to be a trivially derived query (in short, trivial query) if there exist a $M \in L_{left}$ and k tuples $(i_1, u_{i_1}, v_{i_1}), \cdots, (i_k, u_{i_k}, v_{i_k}) \in L_{right}$ such that

- $u_{i_t} = U_t(M, v_0, v_{i_1}, \cdots, v_{i_{t-1}})$ for all $t \in \{1, 2, \cdots, k\}$
- $u = U_{k+1}(M, v_0, v_{i_1}, \cdots, v_{i_k})$

Similarly a left query M is said to be a trivial query if $C^f(M)$ is computable from L_{left}. Any other queries are said to be nontrivial queries.

Definition 3.2.3. BAD Events for Game 0 and Game 1
Let D make q queries to a game (either Game 0 or Game 1). Let u_j be the j^{th} query when it is a right query and M_j be the j^{th} query when it is a left query. For i^{th} left query M_i, let u_i^f be the input to final COM_RO query and $u_{in,1}^i, u_{in,2}^i, \cdots$ be the inputs to the non-final intermediate COM_RO queries. The i^{th} query is said to set the BAD event if one of the following happens

- for nontrivial left query $(M_i, left)$
  - **Collision in final input** The final input is same as final input of a previous left query. $u_i^f = u_j^f; i \neq j$ and $M_i \neq M_j$.
  - **Collision between final and non-final intermediate input**

```
A left query $\mathbf{C}(M)$
    1. $v_0 = IV$.
    2. $\ell = \ell(M)$.
    3. for $i = 1$ to $\ell - 1$
        (a) $u_i = U_i(M, v_0, v_1, \cdots, v_{i-1})$.
        (b) $v_i = COM\_RO(u_i, \lambda, 0)$.
    4. $u_\ell = U_\ell(M, v_0, v_1, \cdots, v_{\ell-1})$.
    5. $v_\ell = COM\_RO(u_\ell, M, 1)$.
    6. return $v_\ell$.

$\mathbf{COM\_RO}(u, M, tag)$
    1. if $tag = 0$ return $f(u)$.
    2. else return $R(M)$

A right query $\mathbf{S}$(u)
    1. If $(j, u, v) \in L$ for some $v, j$, return $v$.
    2. If $\mathcal{P}(L, u) = M \neq \perp$
        (a) $v = COM\_RO(u, M, 1)$.
        (b) $index = index + 1$.
        (c) ADD $(index, u, v)$ to $L$
        (d) return $v$
    3. else $\backslash\backslash \mathcal{P}(L, u) = \perp$
        (a) $v = COM\_RO(u, \lambda, 0)$.
        (b) $index = index + 1$.
        (c) ADD $(index, u, v)$ to $L$
        (d) return $v$
```

Figure 3.3: Procedures of Game 1.

  - $*$ The final input is same as intermediate input of a previous left query, $u_i^f = u_{in,j}^k$ for some $k \leq i$ and $j < l(M_k)$.
  - $*$ One of the intermediate input is same as the final input of a previous left query. $u_{in,k}^i = u_j^f$ for some $j < i$ and $k < l(M_i)$

- **Collision between final input and nontrivial right query** The final input is same as a non-trivial right query u_j; $u_i^f = u_j$ for some $j < i$ but u_j is not a trivial query for M_i.

- for right query $(u_i, right)$
  - **Collision between nontrivial right query and final input of a left query** $u_i = u_j^f$ for some $j < i$ but u_i is not trivially derived.

Let us concentrate on how each of the event defined above can help the distinguisher. When non-trivial collision between the final input of two left (say M_i and M_j) queries happens, the output of two queries will surely be a collision in Game 0. But in case of Game 1, the collision probability will be negligible. When final intermediate input of left query M_i collides with non-final intermediate input of another left query M_j, it may not be obvious how D can exploit this event. But we note that in that case output distribution of these two queries may not be independent in Game 0. The well known length extension attack can also be seen as exploiting this event. Finally if the final input of some left query M_j

collides with input of some nontrivial right query u_i, the outputs of these two queries are same in Game 0. But it is easy to check that, in Game 1, they will be same with negligible probability. We stress that unless the nontrivial right query is same as the final input , adversary cannot gain anything. In fact in both of the games the output distribution remains same, even if the nontrivial right query collides with some non-final intermediate input of some left query.

Theorem 3.2.4. *Let $C \in \mathsf{GDE}$ be a domain extension algorithm. Let* $\textsc{Bad}$ *event be as defined in Definition 3.2.3. Then for any distinguisher D,*

$$|\Pr[D^{C^f,f} = 1] - \Pr[D^{R,S^R} = 1]| \leq \Pr[\textsc{Bad}^{C^f,f}]$$

where $\textsc{Bad}^{C^f,f}$ *denotes the* $\textsc{Bad}$ *event when D is interacting with (C^f, f).*

Proof. To prove the theorem we will show the following relations. Let G_i ($i \in \{0, 1\}$) denote the event that the distinguisher outputs 1 in Game i,

1. $|\Pr[G_0 = 1] - \Pr[G_1 = 1]| \leq \Pr[\textsc{Bad}^0]$.
2. $\Pr[G_1 = 1] = \Pr[D^{R,S^R} = 1]$

As $Pr[D^{C^f,f} = 1] = Pr[G_0 = 1]$ and $\Pr[\textsc{Bad}^0] = \Pr[\textsc{Bad}^{C^f,f}]$, the theorem will follow immediately.

First we shall prove that if $\textsc{Bad}$ events do not happen, then the input output distributions of Game 0 and Game 1 are identical. It is easy to check that $\neg\textsc{Bad}$ is a monotone event as once $\textsc{Bad}$ event happens (flag is set) it remains so for future queries. Now if the $\textsc{Bad}$ events do not happen, then the final input of a left query is always "fresh" in both the games. So the output distribution remains same. On the other hand, if an input to a nontrivial right query is not same as the final input of a previous left query, then in both the cases the outputs are same and the output distribution of the right query is consistent with the previous outputs. Similar to [85], we view each input, output and internal states as random variables. We call the set of input, output and internal states as the transcript of the game. Let T_i^j denote the transcript of Game j after i^{th} query, $j = 0, 1$. Let $\textsc{Bad}_i^0$ and $\textsc{Bad}_i^1$ be the random variable of $\textsc{Bad}$ event in i^{th} query in Game 0 and Game 1 respectively. The following lemma shows that the probability of BAD event occurring first in i^{th} query is same in both Game 0 and Game 1. Moreover if $\textsc{Bad}$ does not happen in first i queries then the transcript after i^{th} query is identically distributed in both the games.

Lemma 3.2.5. *1.* $\Pr[\textsc{Bad}_i^0 \wedge \neg(\cup_{k=1}^{i-1}\textsc{Bad}_k^0)] = \Pr[\textsc{Bad}_i^1 \wedge \neg(\cup_{k=1}^{i-1}\textsc{Bad}_k^1)]$

2. $\Pr[T_i^0 | \neg \cup_{k=1}^{i} \textsc{Bad}_k^1] = \Pr[T_i^1 | \neg \cup_{k=1}^{i} \textsc{Bad}_k^1]$

Corollary 3.2.6. *Let* $\textsc{Bad}^j$ *denote the event that, D invokes* $\textsc{Bad}$ *in Game j. Then we have,*

1. $\Pr[\textsc{Bad}^0] = \Pr[\textsc{Bad}^1]$

2. $\Pr[D^{G_0} \wedge \neg\textsc{Bad}^0] = \Pr[D^{G_1} \wedge \neg\textsc{Bad}^1]$

Proof. Let the distinguisher D makes q queries.So for any $j \in \{0,1\}$ we have

$$\Pr[\text{BAD}^j] = \Pr[\text{BAD}_1^j] + \sum_{i=2}^{q} \Pr[\text{BAD}_i^j \wedge \neg \cup_{k=1}^{i-1} \text{BAD}_k^j].$$

Rest follows directly from Lemma 3.2.5. To prove the second part, check from lemma 3.2.5 that

$$\Pr[D^{G_0} | \neg\text{BAD}^0] = \Pr[D^{G_1} | \neg\text{BAD}^1].$$

Now using the first part we get the result.

□

Using Corollary 3.2.6 one can get the following lemma.

Lemma 3.2.7. *Let G_1 denote the event that the distinguisher outputs* 1 *in Game* 1.

$$|\Pr[G_0 = 1] - \Pr[G_1 = 1]| \leq \Pr[\text{BAD}^0]$$

Proof.

$$\begin{aligned}
\Pr[G_0 = 1] &= \Pr[D^{G_0} = 1] \\
&= \Pr[D^{G_0} \wedge \text{BAD}^0] + \Pr[D^{G_0} \wedge \neg\text{BAD}^0] \\
&= \Pr[D^{G_0} \wedge \text{BAD}^0] + \Pr[D^{G_1} \wedge \neg\text{BAD}^1] \\
&\leq \Pr[\text{BAD}^0] + \Pr[D^{G_1}] \\
&= \Pr[\text{BAD}^0] + Pr[G_1 = 1]
\end{aligned}$$

By symmetry we have

$$\Pr[G_1 = 1] \leq \Pr[\text{BAD}^1] + \Pr[G_1 = 0].$$

Now using Corollary 3.2.6 we get the lemma. □

Now we shall prove that $\Pr[G_1 = 1] = \Pr[D^{R,S^R} = 1]$. We prove it by hybrid arguments.
Game 2: In this game (Fig. 3.5), we change the description of C. Here we remove the lines $1-4$ in the description of C in Game 1 and change the query in line 5 to $COM\_RO(\lambda, M, 1)$ where λ is an empty string. So C does not anymore query COM_RO with $tag = 0$. Note that output of C is still $R(M)$. So the changes does not affect the input output distribution of the game. Hence

$$\Pr[G_2 = 1] = \Pr[G_1 = 1]$$

where G_2 is the event D outputs 1 in Game 2.
Game 3: Now we give S and C a direct access to f and R. So we replace the query $COM\_RO(u, M, 0)$ by $f(u)$. Similarly we write $R(M)$ in place of $COM\_RO(u, M, 1)$. As D did not have direct access to COM_RO and COM_RO did not modify any list, Game 3 is essentially same as Game 2. So

$$\Pr[G_3 = 1] = \Pr[G_2 = 1]$$

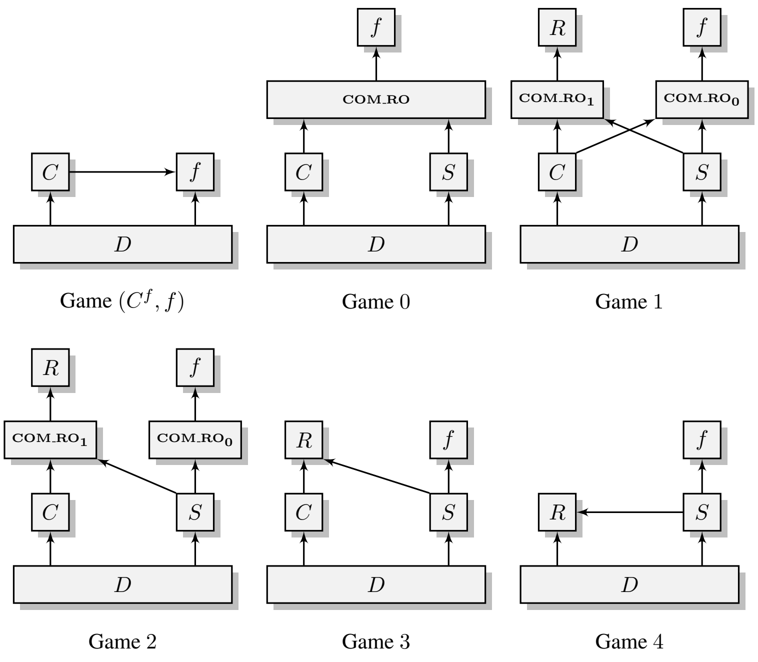

Figure 3.4: Security Games

```
A left query C(M)
    1. $v = COM\_RO(\lambda, M, 1)$.
    2. return $v$.

COM_RO($u, M, tag$)
    1. if $tag = 0$ return $f(u)$.
    2. else return $R(M)$

A right query S(u)
    1. If $(j, u, v) \in L$ for some $v, j$, return $v$.
    2. If $\mathcal{P}(L, u) = M \neq \perp$
        (a) $v = COM\_RO(u, M, 1)$.
        (b) $index = index + 1$.
        (c) ADD $(index, u, v)$ to $L$
        (d) return $v$
    3. else $\backslash\backslash \mathcal{P}(L, u) = \perp$
        (a) $v = COM\_RO(u, \lambda, 0)$.
        (b) $index = index + 1$.
        (c) ADD $(index, u, v)$ to $L$
        (d) return $v$
```

Figure 3.5: Procedures of Game 2.

```
A left query C($M$)
    1. $v = R(M)$.
    2. return $v$.

A right query S(u)
    1. If $(j, u, v) \in L$ for some $v, j$, return $v$.
    2. If $\mathcal{P}(L, u) = M \neq \perp$
        (a) $v = R(M)$.
        (b) $index = index + 1$.
        (c) ADD $(index, u, v)$ to $L$
        (d) return $v$
    3. else \\$\mathcal{P}(L, u) = \perp$
        (a) $v = f(u)$.
        (b) $index = index + 1$.
        (c) ADD $(index, u, v)$ to $L$
        (d) return $v$
```

Figure 3.6: Procedures of Game 3.

where G_3 is the event D outputs 1 in Game 3. Formal description of Game 3 is described in Fig 3.6

Game 4: In this game we remove the subroutine C. So the distinguisher D has direct access to R. Now as the simulator S had no access to internal variables of C, the input output distribution remains same after this change. So

$$\Pr[G_4 = 1] = \Pr[G_3 = 1]$$

where G_4 is the event D outputs 1 in Game 4.

The final observation we make is that S need not query f. Instead it can choose a uniform random value from $\{0,1\}^n$. Note that f is modeled as random function. So we changed a random variable of the game with another random variable of same distribution. Hence all the input, output, internal state distribution remains same. This makes S exactly the same simulator we defined.

$$\Pr[G_4 = 1] = \Pr[D^{R,S^R} = 1].$$

As the Game 0 is equivalent to the pair (C^f, f) we obtain our main result of the section (using triangle inequality):

$$|\Pr[D^{C^f,f} = 1] - \Pr[D^{R,S^R} = 1]| \leq \Pr[\textsc{Bad}^0] = \Pr[\textsc{Bad}^{C^f,f}]$$

$\square$

```
A right query S(u)
  1. If $(j,u,v) \in L$ for some $v, j$, return $v$.
  2. If $\mathcal{P}(L,u) = M \neq \perp$
     (a) $v = R(M)$.
     (b) $index = index + 1$.
     (c) ADD $(index, u, v)$ to $L$
     (d) return $v$
  3. else \\$\mathcal{P}(L,u) = \perp$
     (a) $v = \leftarrow_R \{0,1\}^n$.
     (b) $index = index + 1$.
     (c) ADD $(index, u, v)$ to $L$
     (d) return $v$
```

Figure 3.7: The Generic Simulator.

3.3 Proof of Lemma 3.2.5

Let $X_1^j, X_2^j, \cdots, X_q^j \in \mathcal{X}$ and $Y_1^j, Y_2^j, \cdots, Y_q^j \in \mathcal{Y}$ be input random variables and output random variables respectively of Game j; $j \in \{0,1\}$. Let $U_{1,i}^j, U_{2,i}^j, \cdots, U_{\ell_i,i}^j$ be the internal random variables (output of internal queries) of i^{th} query in Game j. As previously We call the set of input,output and internal states, the transcript of the game. Let T_i^j denote the transcript of Game j after i^{th} query.

In this proof, w.l.g., we assume that Distinguisher does not repeat queries. Let q be the number queries the adversary makes. We shall prove the Lemma 3.2.5 by induction on i.

CASE $i = 1$: We start from the observation that

$$\Pr_D[X_1^0] = \Pr_D[X_1^1].$$

If X_1 is a left query $(M_1, left)$

$$\Pr_{D,f}[(U_1^0, \cdots, U_{\ell_1,1}^0)|X_1^0] = \Pr_{D,f}[(U_1^1, \cdots, U_{\ell_1,1}^1)|X_1^1].$$

Hence

$$\Pr_{D,f}[X_1^0, U_{1,1}^0, U_{2,1}^0, \cdots, U_{\ell_1,1}^0] = \Pr_{D,f,R}[X_1^1, U_{1,1}^1, U_{2,1}^1, \cdots, U_{\ell_1,1}^1].$$

It is easy to check that for the first query $\text{B}\textsc{ad}$ event can only be set by a left query. Also note that it happens when the final query is same with some non-final intermediate query. So

$$\Pr_{D,f}[\textsc{Bad}^0_1|X^0_1, U^0_{1,1}, U^0_{2,1}, \cdots, U^0_{\ell_1,1}] = \Pr_{D,R,f}[\textsc{Bad}^1_1|X^0_1, U^0_{1,1}, U^0_{2,1}, \cdots, U^0_{\ell_1,1}].$$

Hence

$$\Pr_{D,f}[\textsc{Bad}^0_1] = \Pr_{D,R,f}[\textsc{Bad}^1_1]$$

If $\neg\textsc{Bad}_1$ is true then $u^f_1 \notin I_{M_1}$. As f and R are random oracles, we have

$$\Pr_f[f(u^f_1) = v] = \Pr_R[R(M_1) = v] \forall v \in \{0,1\}^n.$$

On the other hand if the first query is $(u, right)$ for any u, then $Y_1 = f(u)$ in both the games. So, $\forall v \in \{0,1\}^n$

$$\Pr_{D,f}[Y^0_1 = v|X^0_1, U^0_{1,1}, U^0_{2,1}, \cdots, U^0_{\ell_1,1} \wedge \neg\textsc{Bad}^0_1]$$
$$= \Pr_{D,R,f}[Y^1_1 = v|X^1_1, U^1_{1,1}, U^1_{2,1}, \cdots, U^1_{\ell_1,1} \wedge \neg\textsc{Bad}^1_1].$$

Hence,

$$\Pr_{D,f}[X^0_1, U^0_{1,1}, U^0_{2,1}, \cdots, U^0_{\ell_1,1}, Y^0_1|\neg\textsc{Bad}^0_1]$$
$$= \Pr_{D,f,R}[X^1_1, U^1_{1,1}, U^1_{2,1}, \cdots, U^1_{\ell_1,1}, Y^0_1|\neg\textsc{Bad}^1_1].$$

This implies that the distribution of transcript after first query is identical in both the games if $\neg\textsc{Bad}_1$ is true. This finishes the proof of the case $i = 1$.
Suppose the lemma is true for all $i < t$.
Case $i = t$:By Induction Hypothesis, we have,

$$\Pr_{D,f}[T^0_{t-1}|\neg(\cup^{t-1}_{k=1}\textsc{Bad}^0_k)] = \Pr_{D,f,R}[T^1_{t-1}|\neg(\cup^{t-1}_{k=1}\textsc{Bad}^1_k)].$$

As the input/output distribution of two games are same if $\neg(\cup^{t-1}_{k=1}\textsc{Bad}^1_k)$ is true, the distribution of t^{th} query must be same for both the games.

$$\Pr_{D,f}[X^0_t|T^0_{t-1} \wedge \neg(\cup^{t-1}_{k=1}\textsc{Bad}^0_k)] = \Pr_{D,f,R}[X^1_t|T^1_{t-1} \wedge \neg(\cup^{t-1}_{k=1}\textsc{Bad}^1_k)]$$

When $X_t = (u_t, right)$ is a non trivial right query then $Y_t = f(u_t)$ in both the games. Now if $\neg(\cup^{t-1}_{i=1}\textsc{Bad}_i)$ is true then, from induction hypothesis, the transcript distribution after t queries is same for both the games. The probability that u^f_t collides with some final input of any previous query is same for both the games. So for the right query

$$\Pr[\textsc{Bad}^0_t|\neg(\cup^{t-1}_{k=1}\textsc{Bad}^0_k)] = \Pr[\textsc{Bad}^1_t|\neg(\cup^{t-1}_{k=1}\textsc{Bad}^1_k)]$$

When $X_t = (M_t, left)$ then we have,

$$\Pr_{D,f}[X_t^0 | T_{t-1}^0 \wedge \neg\textsc{Bad}^0] = \Pr_{D,f,R}[X_t^1 | T_{t-1}^1 \wedge \neg\textsc{Bad}^1]$$

Notice that if the distribution of t^{th} query is same for both the games then the distribution of internal queries is also same for both the games. Hence

$$\Pr_{D,f}[X_t^0, U_{1,1}^0, \cdots, U_{\ell(M_t),(t)}^0 | T_{t-1}^0 \wedge \neg(\cup_{k=1}^{t-1}\textsc{Bad}_k^0)]$$
$$= \Pr_{D,f}[X_t^1, U_{1,1}^1, \cdots, U_{\ell(M_t),(t)}^1 | T_{t-1}^1 \wedge \neg(\cup_{k=1}^{t-1}\textsc{Bad}_k^1)].$$

Hence

$$\Pr_{D,f}[\textsc{Bad}_t^0 | \neg(\cup_{k=1}^{t-1}\textsc{Bad}_k^0)] = \Pr_{R,D,f}[\textsc{Bad}_t^1 | \neg(\cup_{k=1}^{t-1}\textsc{Bad}_k^1)]$$

For a non-trivial right query $(u_t, right)$, both the games query $f(u_t)$. If $\neg(\cup_{k=1}^{t}\textsc{Bad}_k^1)$ is true then $u_t \neq u_j^f$ for all $j < t$. On the other hand , for a left query $(M_t, left)$, if $\neg(\cup_{k=1}^{t}\textsc{Bad}_k^1)$ is true then u_t^f has never been queried before. Then $\Pr_f[f(u_t^f) = v] = \Pr_R[R(M_t) = v]$ for all $v \in \{0,1\}^n$. So

$$\Pr_{D,f}[Y_t^0, X_t^0, U_{1,1}^0, \cdots, U_{\ell(M_t),(t)}^0 | T_{t-1}^0 \wedge \neg(\cup_{k=1}^{t}\textsc{Bad}_k^0)]$$
$$= \Pr_{D,R,f}[Y_t^1, X_t^1, U_{1,1}^1, \cdots, U_{\ell(M_t),(t)}^1 | T_{t-1}^1 \wedge \neg(\cup_{k=1}^{t}\textsc{Bad}_k^1)].$$

This implies

$$\Pr_{D,f}[T_t^0 | \neg(\cup_{k=1}^{i}\textsc{Bad}_k^0)] = \Pr_{D,R,f}[T_t^1, \cdots, U_{\ell_i,i}^1) | \neg(\cup_{k=1}^{i}\textsc{Bad}_k^1)]$$

□

3.4 Applications to popular mode of operations

In this section we show the indifferentiability of different popular mode of operations from a Random Oracle. We note that, according to Theorem 3.2.4 to upper bound distinguisher's advantage one needs to calculate the probability of BAD event defined in previous section. Moreover we can only concentrate on the specific mode of operation rather than the output of the simulator.

3.4.1 Merkle-Damgård with HAIFA

Now we consider Merkle-Damgård mode of operation another variant of prefix free padding; HAIFA. In this padding we append a counter (indicating the block number) with each but last block of the message. The last block is padded with 0 (see Fig 3.8). It is easy to check that Merkle-Damgård with HAIFA

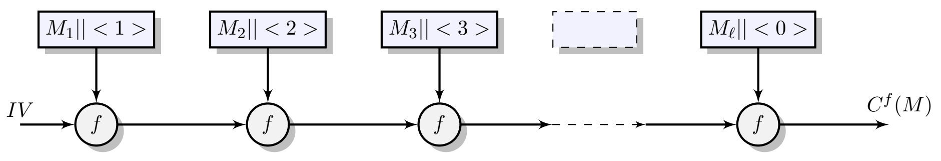

Figure 3.8: Merkle-Damgård with padding rule HAIFA

belongs to GDE. In this case the reconstruction algorithm works as follows. Let t denote the length of the padding. On input of a f query u; check whether the last t bit of u is 0. If not return $\perp$. Otherwise parse u as $h_0||m_0$ where h_0 is of n bits. Find, whether h_0 is in the output column of a query in the list L. If no return $\perp$. If such a query exists select corresponding input u_i. Now last t bit of u_i will be $\ell - 1$, where ℓ is the number of blocks in possible message. We call such an u_i as $u_{\ell-1}$. Now for $j = \ell - 1$ to 2; parse u_j as $h_{j-1}||m_j$. find whether h_{j-1} exist in the output column of L where the corresponding input has padding $j - 1$. If no return $\perp$. Else select the input and call it u_{j-1}. Repeat the above three steps until we find a u_j with padding 1. If we can find such u_is, then construct the message $M = m_1||m_2|| \cdots m_\ell||m_0$ and return M. Check that for i^{th} query the algorithm P runs in time $\mathcal{O}(i\ell)$ where ℓ is the maximum block length of a query. Hence the total running time of P and hence of the simulator is $\mathcal{O}(q^2\ell)$.

For finding the probability of BAD events, the HAIFA padding rule gives us the following advantage. While computing $C^f(M)$ for any message M, all the intermediate inputs are unique. In fact the final input is always different from any intermediate input. So if no f query with same counter padding has collision in the output, the outputs of the penultimate f queries do not have collision in output and no nontrivial right query input is same as the final input of some left query, BAD event does not happen. If BAD event does not happen in i^{th} query, the output of i^{th} query is uniformly distributed over $Y = \{0,1\}^n$. Without loss of generality, we assume that D does not make any trivial query as trivial queries do not raise a BAD event. Moreover we can consider only a deterministic (albeit adaptive) distinguisher as the general case can easily be reduced to this case [89]. So input to the i^{th} query is uniquely determined by previous $i - 1$ outputs. We represent the output of the nontrivial queries as the view (V) of the distinguisher. Let $f : \{0,1\}^{m'} \to \{0,1\}^n$ be a fixed input length random oracle. If D makes q nontrivial queries and $\mathcal{V}$ is the set of all possible views then $|\mathcal{V}| = |Y|^q$. We write V as $\cap_{i=1}^q V_i$, where V_i is the output corresponding to i^{th} query. Now for any $V \in \mathcal{V}$, we define an event BAD'^V which occurs whenever there is a collision between intermediate inputs, final inputs and right query inputs. In fact, $\neg\text{BAD}'^V \cap V \subseteq \neg\text{BAD} \cap V$. We split, BAD'^V as $\cup_{i=1}^q \text{BAD}_i'^V$. $\text{BAD}_i'^V$ occurs whenever any intermediate input (final or non-final) of i^{th} left query collides with any intermediate inputs of any other distinct left query or with input of any nontrivial right query. Although we are working with an adaptive attacker, future query inputs are fixed by V. Note that, if i^{th} query is right query $\text{BAD}_i'^V$ never occurs. Suppose ℓ_i is the number of blocks in i^{th} query.

Suppose the i^{th} query made by the distinguisher is a left query. For $\neg\text{BAD}_i'^V$ to happen, any intermediate input (final or non-final) has to be different from previous intermediate/final inputs. Because of HAIFA padding, no final input will be same with any intermediate input. So if $\neg\text{BAD}_i'^V$ has to be true, every intermediate input of i^{th} has to be different from the intermediate inputs with same counter of previous $i-1$ queries. Also any intermediate input cannot be same as future right query inputs or future left query intermediate inputs fixed by the view. There only q many such candidates. So for any intermediate(final) input there are at most $i-1+q < 2q$ bad values. Hence,

$$\Pr[\neg\text{BAD}_i'^V \cap V_i | \cap_{j=1}^{i-1} (\neg\text{BAD}_j'^V \cap V_j)] \geq \left(\frac{|Y|-2q}{|Y|}\right)^{\ell_i-1} \cdot \frac{1}{|Y|}.$$

If the i^{th} query is nontrivial right query,

$$\Pr[\neg\text{BAD}_i'^V \cap V_i | \cap_{j=1}^{i-1} (\neg\text{BAD}_j'^V \cap V_j)] = \frac{1}{|Y|}.$$

So one can calculate the probability of $\neg\text{BAD}$ as

$$\begin{aligned}
\Pr[\neg\text{BAD}] &= \sum_{V\in\mathcal{V}} \Pr[\neg\text{BAD} \cap V] \geq \sum_{V\in\mathcal{V}} \Pr[\neg\text{BAD}'^V \cap V] \\
&= \sum_{V\in\mathcal{V}} \Pr[\cap_{i=1}^{q}(\neg\text{BAD}_i'^V \cap V_i)] \\
&= \sum_{V\in\mathcal{V}} \Pr[\neg\text{BAD}_1'^V \cap V_1] \prod_{i=2}^{q} \Pr[\neg\text{BAD}_i'^V \cap V_i | \cap_{j=1}^{i-1} (\neg\text{BAD}_j'^V \cap V_j)] \\
&\geq \sum_{V\in\mathcal{V}} \prod_{i=1}^{q} \left(\frac{|Y|-2q}{|Y|}\right)^{\ell_i-1} \cdot \frac{1}{|Y|} \\
&\geq \sum_{V\in\mathcal{V}} \left(1 - \mathcal{O}(\frac{\sigma q}{|Y|})\right) \cdot \frac{1}{|Y|^q} = 1 - \mathcal{O}(\frac{\sigma q}{|Y|})
\end{aligned}$$

Here $Y = \{0,1\}^n$ and $\sigma = \sum_{i=1}^{q} \ell_i$. So $\Pr[\text{BAD}] \leq \mathcal{O}(\frac{\sigma q}{2^n})$.

Theorem 3.4.1. *The Merkle-Damgård construction with HAIFA padding rule based on a FIL-RO is $(t_S, q_C, q_{\mathcal{F}}, \varepsilon)$ - indifferentiable from a random oracle, with $t_S = \ell \cdot \mathcal{O}(q^2)$ and $\varepsilon = \mathcal{O}(\frac{\sigma q}{2^n})$, where ℓ is the maximum length of a query made by the distinguisher D, σ is the sum of the lengths of the queries made by the distinguisher and $q = q_C + q_{\mathcal{F}}$.*

In [36], Coron et al. considered a specific prefix-free padding rule which is similar to HAIFA. There they proved indifferentiability bound as $\mathcal{O}(\frac{\sigma^2}{2^n})$. So Theorem 3.4.1 can be seen as improving that bound as well. In Section 3.5.1 we show that the bound we prove in Theorem 3.4.1 is tight.

3.4.2 Tree Mode of Operation with counter

Tree mode of operation is another popular mode of operation. *MD6*, a SHA3 candidate uses this mode of operation. Let $f : \{0,1\}^{m'} \to \{0,1\}^n$. The input message is divided in blocks and can be viewed as the leaf nodes. The edges are the function f. Any internal node can be viewed as the concatenation of the outputs of f on its child nodes. The output of the hash function is the output of f applied on the root. Now with each node we associate a tag $\langle height, index \rangle$ where $height$ denotes the height of the

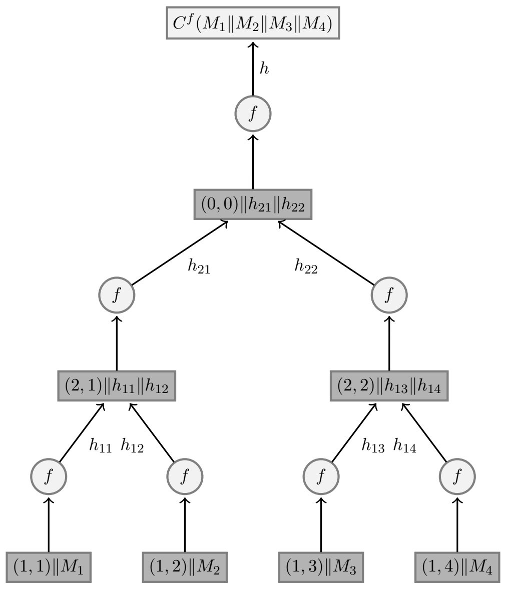

Figure 3.9: Tree Mode of Operation with Sequential Padding where $\frac{m'}{n} = 2$

node in the tree and $index$ represents the index of the node in the level it is in (see Figure 3.9). Each node is padded with the tag. This padding makes, like HAIFA, each input unique in the evaluation tree of $C^f(M)$ for any fixed message M. In [44], Dodis et al. proved the tree mode to be indifferentiable with $\mathcal{O}(\frac{q^2\ell^2}{2^n})$ bound. In this section we improve their bound to $\epsilon = \mathcal{O}(\frac{q^2 \log \ell}{2^n})$.

One can easily construct the computable algorithm $\mathcal{P}$ using the same method as in HAIFA. Let M_i and M_j be two distinct left queries (for simplicity, both of length ℓ) made by distinguisher. Let k be an index such that k^{th} blocks of M_i and M_j are different. Consider the path from node $(1,k)$ to the root. It is easy to check that if no collision happens in this path, the final input of f query does not collide while computing $C^f(M_i)$ and $C^f(M_j)$. Length of this path is $\log \ell$ (height of the tree). On the other hand a nontrivial right query input can collide with at most one intermediate input of a left query. Hence, using a method similar to proof of Theorem 3.4.1, one can prove the following theorem

Theorem 3.4.2. *Let $\mathcal{F}$ be a FIL-RO and C be the tree mode of operation with the counter padding. $C^{\mathcal{F}}$ is $(t_S, q_C, q_{\mathcal{F}}, \varepsilon)$ - indifferentiable from a random oracle, with $t_S = \ell \cdot \mathcal{O}(q^2)$ and $\varepsilon = \mathcal{O}(\frac{q^2 \log \ell}{2^n})$, where ℓ is the maximum length of a query made by the distinguisher D and $q = q_C + q_{\mathcal{F}}$.*

Proof. As in Section 3.4.1, we define an event BAD'^V, for each $V \in \mathcal{V}$. But as discussed in Section 3.4.2 we do not really need all intermediate inputs with same counter to be distinct. All we need is, all the intermediate inputs are distinct in at least one path for every pair of left queries. So BAD'_i does not happen if there exist paths $P_{i,j}$ from a leaf node to final input node in the derivation tree of i^{th} query such that all inputs along that path did not appear in j^{th} query; $j < i$. Also, as before BAD'^V_i happens if any intermediate input or final input collides with future right query inputs or fixed intermediate inputs of a left query. As before, we shall give lower bound of $\Pr[\neg\text{BAD}'^V_i \cap V_i | \cap_{j=1}^{i-1} (\neg\text{BAD}'^V_j \cap V_j)]$. At least one input block in every pair of distinct left queries (i, j) is different. For simplicity we assume the input with pad $(1, 1)$ is different in all the i queries. The general case can be handled in a same way, but the proof will be a bit more messy.

Let P denote the path from $(1, 1)$ to root. For each intermediate input along the path P, there are at most $(i - 1)$ previous intermediate inputs which has same padding. Also, as each intermediate input has unique pad, any particular right query inputs can collide with at most one intermediate input of the i^{th} left query. Let $b_{(k,k')}$ denote the number of right queries with pad (k, k') . $\sum_{(k,k')} b_{(k,k')} \leq q$. So for $(k, k')^{\text{th}}$ node in the path P, there are $i - 1 + b_{(k,k')} < 2q$ possible values which will set BAD'^V_i . For any other node there are $b_{(k,k')}$ such values. Hence, if i^{th} query is a left query we have,

$$\begin{aligned}
&\Pr[\neg\text{BAD}'^V_i \cap V_i|\cap_{j=1}^{i-1}(\neg\text{BAD}'^V_j \cap V_j)] \\
&\qquad\geq \left(\frac{|Y| - 2q}{|Y|}\right)^{\log \ell} \cdot \left(\prod_{(k,k')\notin P} \frac{|Y| - b_{(k,k')}}{|Y|}\right) \cdot \frac{1}{|Y|} \\
&\qquad\geq \left(1 - \frac{\mathcal{O}(q \log \ell)}{|Y|}\right) \cdot \left(1 - \frac{q}{|Y|}\right) \cdot \frac{1}{|Y|} \\
&\qquad= \left(1 - \frac{\mathcal{O}(q \log \ell)}{|Y|}\right) \cdot \frac{1}{|Y|}
\end{aligned}$$

If the i^{th} query is nontrivial right query,

$$\Pr[\neg\text{BAD}'^V_i \cap V_i| \cap_{j=1}^{i-1} (\neg\text{BAD}'^V_j \cap V_j)] = \frac{1}{|Y|}.$$

Hence,

$$\begin{aligned}
\Pr[\neg\text{BAD}] &\geq \sum_{V\in\mathcal{V}} \prod_{i=1}^{q} \left(1 - \frac{\mathcal{O}(q \log \ell)}{|Y|}\right) \cdot \frac{1}{|Y|} \\
&= \sum_{V\in\mathcal{V}} \left(1 - \frac{\mathcal{O}(q^2 \log \ell)}{|Y|}\right) \cdot \frac{1}{|Y|^q} \\
&= 1 - \frac{\mathcal{O}(q^2 \log \ell)}{|Y|}
\end{aligned}$$

□

3.5 Indistinguishability attacks on popular mode of operations

In this section we show a lower bound for the advantage of a distinguishing attacker against Merkle-Damgård constructions with HAIFA padding and Tree mode of operations with counter padding scheme. The bound we achieve actually reaches the corresponding upper bound shown before. Note, if all the queries are of length ℓ, then $q^2\ell = q\sigma$.

3.5.1 Distinguishing Attacks on Merkle-Damgård Constructions

Consider q messages $M_1, \cdots, M_q$ such that,

$$\begin{aligned}
\textsc{Pad}(M_1) &= M_1^1 || M^2 || \cdots || M^\ell \\
\textsc{Pad}(M_2) &= M_2^1 || M^2 || \cdots || M^\ell \\
&\vdots \\
\textsc{Pad}(M_q) &= M_q^1 || M^2 || \cdots || M^\ell
\end{aligned}$$

Let $\textsc{Coll}$ be the event denoting collision among $C^f(M_1), \cdots, C^f(M_q)$. We shall prove that,

$$\Pr[\textsc{Coll}] = \Omega(\frac{q^2\ell}{2^n}).$$

Let $\textsc{Coll}_{ij}$ be the event denoting the collision between $C^f(M_i)$ and $C^f(M_j)$. Hence,

$$\Pr[\textsc{Coll}] = \Pr[\bigcup_{1\le i<j\le q} \textsc{Coll}_{ij}].$$

Using principle of inclusion-exclusion we get,

$$\begin{aligned}
&\Pr[\bigcup_{1\le i<j\le q} \textsc{Coll}_{ij}] \ge \sum_{1\le i<j\le q} \Pr[\textsc{Coll}_{ij}] - \sum_{1\le i<j<k\le q} \big(\Pr[\textsc{Coll}_{ij} \cap \textsc{Coll}_{jk}] \\
&+ \Pr[\textsc{Coll}_{ij} \cap \textsc{Coll}_{ik}] + \Pr[\textsc{Coll}_{ik} \cap \textsc{Coll}_{jk}]\big) \\
&- \sum_{1\le i<j<k<r\le q} \big(\Pr[\textsc{Coll}_{ij} \cap \textsc{Coll}_{kr}] + \Pr[\textsc{Coll}_{ik} \cap \textsc{Coll}_{jr}] \\
&+ \Pr[\textsc{Coll}_{ir} \cap \textsc{Coll}_{jk}]\big) \qquad (3.1)
\end{aligned}$$

Next, we prove the following Lemma.

Lemma 3.5.1. *Let* $Y = \{0,1\}^n$ *and* $1 \le i < j < k < r \le q$. *If* $\ell - 1 \le 2^n$, *then*

1. $\Pr[\textsc{Coll}_{ij}] \ge \frac{\ell}{2|Y|}$
2. $\Pr[\textsc{Coll}_{ij} \cap \textsc{Coll}_{jk}] \le \frac{2\ell^2}{|Y|^2}$
3. $\Pr[\textsc{Coll}_{ij} \cap \textsc{Coll}_{kr}] \le \frac{\ell^2}{|Y|^2} + \frac{6\ell^3}{|Y|^3}$

Proof. Proof of part 1
Let $Y = \{0,1\}^n$. The probability of i^{th} left query and j^{th} left query will collide is lower bounded as following.

$$\begin{aligned}
\Pr[\textsc{Coll}_{ij}] &= \frac{1}{|Y|} + (1-\frac{1}{|Y|})\frac{1}{|Y|} + (1-\frac{1}{|Y|})^2\frac{1}{|Y|} + \cdots + (1-\frac{1}{|Y|})^{\ell-1}\frac{1}{|Y|} \\
&= 1-(1-\frac{1}{|Y|})^{\ell} \\
&\geq \frac{\ell}{2|Y|}
\end{aligned}$$

□

Proof of part 2

$$\begin{aligned}
&\Pr[\textsc{Coll}_{ij} \cap \textsc{Coll}_{jk} | \textsc{Coll}_{ij} \wedge \textit{ First time collision happened on input of} \\
&\quad m^{k'} \textit{ in } i^{th} \textit{ and } j^{th} \textit{ queries}] \\
&= \frac{2}{|Y|}(1+(1-\frac{2}{|Y|}) + \cdots + (1-\frac{2}{|Y|})^{k'-2}) + \frac{1}{|Y|}(1-\frac{2}{|Y|})^{k'-1}(1+(1-\frac{1}{|Y|}) \\
&\quad + \cdots + (1-\frac{1}{|Y|})^{\ell-k'}) \\
&\leq \frac{2}{|Y|}(1+(1-\frac{1}{|Y|}) + \cdots + (1-\frac{1}{|Y|})^{\ell-1}) \\
&= 2(1-(1-\frac{1}{|Y|})^{\ell}) \\
&\leq \frac{2\ell}{|Y|}
\end{aligned}$$

The above result actually implies $\Pr[\textsc{Coll}_{ij} \cap \textsc{Coll}_{jk} | \textsc{Coll}_{ij}] \leq \frac{2\ell}{|Y|}$. Hence,

$$\begin{aligned}
\Pr[\textsc{Coll}_{ij} \cap \textsc{Coll}_{jk}] &= \Pr[\textsc{Coll}_{ij}] \times \Pr[\textsc{Coll}_{ij} \cap \textsc{Coll}_{jk} | \textsc{Coll}_{ij}] \\
&\leq (1-(1-\frac{1}{|Y|})^{\ell}) \times \frac{2\ell}{|Y|} \\
&\leq \frac{2\ell^2}{|Y|^2}
\end{aligned}$$

□

Proof of Part 3
With a similar approach to the above we can show

$$\Pr[\textsc{Coll}_{ij} \cap \textsc{Coll}_{kr} | \textsc{Coll}_{ij}] \leq \frac{\ell}{|Y|} + \frac{6\ell^2}{|Y|^2}.$$

Hence,

$$\Pr[\textsc{Coll}_{ij} \cap \textsc{Coll}_{kr}] \leq \frac{\ell^2}{|Y|^2} + \frac{6\ell^3}{|Y|^3}.$$

□

Using Equation 3.1 and Lemma 3.5.1 we get,

$$\Pr[\textsc{Coll}] \geq \binom{q}{2}\frac{\ell}{2|Y|} - 3\binom{q}{3}\frac{2\ell^2}{|Y|^2} - 3\binom{q}{4}(\frac{\ell^2}{|Y|^2} + \frac{6\ell^3}{|Y|^3}) \approx \frac{\alpha}{4} - \frac{\alpha^2}{8} \geq \frac{\alpha}{8}$$

where $\alpha = \frac{q^2\ell}{2^n} < 1$. By Birthday Bound, for a random function R, the collision probability for q different messages is $\Theta(\frac{q^2}{2^n})$. Hence for a distinguisher D which queries messages $M_1, \cdots, M_q$, the advantage of the distinguisher is $\Omega(\frac{q^2\ell}{2^n})$. Also we can easily construct q such messages for any prefix free Merkle-Damgård scheme, specifically HAIFA.

Theorem 3.5.2. *Let C be the Merkle-Damgård domain extension with a prefix free padding. There exists a distinguisher D, such that*

$$|\Pr[D^{C^f,f} = 1] - \Pr[D^{S^R,R} = 1]| \geq \Omega(\frac{q^2\ell}{2^n})$$

where D makes q queries and length of each query is at most ℓ.

3.5.2 Distinguishing Attacks on Tree Mode

Similar to previous attack we choose q messages $M_1, \cdots, M_q$ such that after padding only first block of these messages are different. Formally

$$\textsc{Pad}(M_i) = M_i^1||M^2||\cdots||M^\ell.$$

Now for these massages the tree mode works like a Merkle-Damgård mode with messages $\overline{M}_1, \cdots, \overline{M}_q$ where

$$\textsc{Pad}(\overline{M_i}) = M_i^1||\overline{M}^2||\cdots||\overline{M}^h \; \forall i = 1, 2, \cdots, q$$

$h = \lceil \log \ell \rceil$ is the height of the tree. Hence using the similar method to previous section we get the following Theorem.

Theorem 3.5.3. *Let C be the Tree mode domain extension with the sequential counter padding. There exists a distinguisher D, such that*

$$|\Pr[D^{C^f,f} = 1] - \Pr[D^{S^R,R} = 1]| \geq \Omega(\frac{q^2 \log \ell}{2^n})$$

where D makes q queries and length of each query is at most ℓ.

Chapter 4

Indifferentiability of JH Domain Extension

4.1 Introduction

In 2007, NIST announced a competition for a new hash function standard, to be called SHA-3. 64 designs were submitted and after an internal review of the submissions, 51 were selected for meeting the minimum submission requirements and accepted as the First Round Candidates. Recently, NIST declared the names of 14 candidates for the second round of the competition. One of these candidates will win the competition and eventually become the next standard cryptographic hash function. Hence, it is essential for these candidate designs to meet the state of the art security notions.

In this chapter, we consider the mode of operation of the JH hash function, one of the final round candidates of SHA3 competition. It uses a novel construction, somewhat reminiscent of a sponge construction [15], to build a hash algorithm out of a single, large, fixed permutation using chopped-MD domain extension [113]. We also consider a little modified mode of operation of JH where the chopping is done on the other bits.

Contribution of this chapter

In this chapter we examine the indifferentiability and preimage resistance of JH mode of operation in $2n$ bit random permutation model. Let s denote the number of chopped bits. We extend the technique of Chang and Nandi [32] to random permutation model. We prove that under the assumption that the fixed permutation of JH is a random permutation, JH mode of operation with specific length padding is indifferentiable from random oracle with distinguisher's advantage bounded by $O(\frac{q^2\sigma}{2^s} + \frac{q^3}{2^n})$. When $s = 3n/2$ (as in case of JH hash function with 256 bit output), our result gives beyond the birthday barrier security guarantee for JH [1]. This implies that finding collision in the output is not enough to distinguish a Random Oracle from JH hash function with $n/2$-bit output. Although chopMD constructions do not need the length padding in general, we show the padding is *essential* for JH mode. We construct one indifferentiability attacker, working in *constant* number of queries against JH mode of operation without length padding at last block with n-bit output. This result also shows that the method used in [15] to prove indifferentiability of sponge constructions (where length padding in last block is not required) based on random permutations cannot be readily extended to prove indifferentiability of JH.

Simultaneously, we look at other constructions, modifying JH mode of operation, where the chopping is done on the first instead of last s bits.

- We show that when the length of longest query is less than $2^{n/2}$, then the modified JH mode of operation without the length padding is indifferentiable from an RO with distinguisher's advantage bounded by $\mathcal{O}(\frac{q^2}{2^{\min(s,n)}})$ where q is the maximum number of queries made by the distinguisher.
- We show one indifferentiability attacker against modified JH mode of operation with $\Omega(2^{n/2})$ query complexity. This shows for $s \geq n$ the previous security bound is actually optimal.
- If we set $s = n$, we get a random permutation based secure mode of operation with n-bit digest using $2n$ bit permutation. We note that this construction is at least as secure as the sponge construction based on $2n$ bit random permutation. where the indifferentiability bound is $\mathcal{O}(\frac{\sigma^2}{2^n})$ [15]. Here σ is the number blocks that the adversary queries.

Remark 4.1.1. Although, for $s = n$, the bound of JH' is optimal, for $s < n$, the bound of JH$'$ may not be optimal.

Overview of our technique

We extend and formalize the Technique of [32] to the permutation based constructions. Informally speaking, we avoid the cumbersome game playing arguments. We show that, in order to prove indifferentiability one has to construct a simulator, such that the statistical distance of the distribution (of the view of any indifferentiability distinguisher) between the ideal and real world is negligible. Then we use the technique of Interpolation Probability to show that for our simulator, for any output view. the statistical distance between the real and the ideal world is negligible. Moreover, our simulator does not invoke any BAD flag. Instead, the simulator goes through a resampling procedure to ensure that the distribution remains close to the real world.

[1]According to birthday paradox, for a uniform random function with n-bit digest, collision can be found with significant probability in $\mathcal{O}(2^{\frac{n}{2}})$ queries. This is known as the birthday barrier as security against more than $\mathcal{O}(2^{\frac{n}{2}})$ queries is non-trivial; when at all possible

4.2 Preliminaries

Definition 4.2.1. $FH : \{0,1\}^{2n} \rightarrow \{0,1\}^n$ is a function which outputs first n bit of any $2n$ bit number. Similarly, $LH : \{0,1\}^{2n} \rightarrow \{0,1\}^n$ is a function which outputs last n bit of any $2n$ bit number.

Often we refer FH as left half and LH as right half. Below we state a few basic inequalities as a lemma which will be useful later.

Lemma 4.2.2. *For any $y \in \{0,1\}^{2n-s}$, $c \in \{0,1\}^n$, $\mathcal{S} \subseteq \{0,1\}^{2n}$ and $\mathcal{T} \subseteq \{0,1\}^n$ we have,*

1. $|\{z \in \{0,1\}^s : y\|z \in \mathcal{S}\}| \leq |\mathcal{S}|$ *and* $|\{z \in \{0,1\}^s : z\|y \in \mathcal{S}\}| \leq |\mathcal{S}|$
2. $|\{z \in \{0,1\}^s : FH(y\|z) \oplus c \in \mathcal{T}\}| \leq 2^n|\mathcal{T}|$ *and* $|\{z \in \{0,1\}^s : FH(z\|y) \oplus c \in \mathcal{T}\}| \leq \frac{2^s}{2^{\min(s,n)}}|\mathcal{T}|$

JH Compression Function

The compression function of JH, $f^\pi : \{0,1\}^{3n} \rightarrow \{0,1\}^{2n}$ is defined as follows:

$$f^\pi(h_1\|h_2\|m) = \pi(h_1\|(h_2 \oplus m)) \oplus (m\|0^n)$$

where $h_1, h_2, m \in \{0,1\}^n$ and $\pi : \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$ is a fixed permutation.

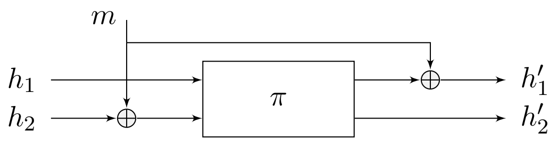

Figure 4.1: The JH compression function

JH Mode of Operation

The JH mode of operation based on a permutation π is the chopMD mode of operation based on the above compression function f^π. The usual Merkle-Damgård technique is applied on f^π and the output of the hash function is the first $2n-s$ bits of the final f^π query output. For any, $0 \leq s \leq |n|$, $\text{CHOPR}_s(m)$ is defined as m_L where $m = m_L\|m_R$ and $|m_R| = s$. Formally the JH mode of operation based on a permutation π with initial value $IV_1\|IV_2$ is defined as

$$JH^\pi(\cdot) : (\{0,1\}^n)^+ \rightarrow \{0,1\}^{2n-s} \equiv \text{CHOPR}_s(MD^{f^\pi}(\cdot)).$$

Where, MD^{f^π} is the Merkle-Damgård mode of operation with initial value as $IV_1\|IV_2$ and compression function as f^π. According to [113], typically $s = n$. Also it is suggested to have $s \geq n$.

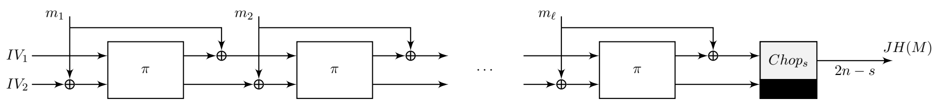

Figure 4.2: The JH hash function

Modified JH Mode of Operation: JH'

We also define a modified version of JH mode of operation (denoted by JH' throughout the chapter) where instead of chopping right most s bits we chop left most s bits. Let for $0 \le s \le |m|$, $\textsc{ChopL}_s(m)$ is defined as m_R where $m = m_L \| m_R$ and $|m_L| = s$.

$$JH'^{\pi}(\cdot) : (\{0,1\}^n)^+ \to \{0,1\}^{2n-s} \equiv \textsc{ChopL}_s(MD^{f^\pi}(\cdot)).$$

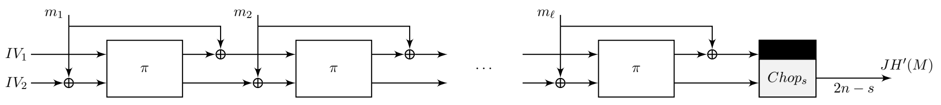

Figure 4.3: The JH' hash function

Throughout the chapter JH-t denotes the JH mode of operation with t bit output. Similarly JH'-t denotes JH' mode of operation with t bit output.

`Padding Rule`: To encode messages whose lengths are not multiple of block size (n bit) we need some padding rule, so that padded message becomes a multiple of block size. A simple padding rule can be zero padding, that is adding sufficient number of zero bits so that the padded message becomes a multiple of block size, even though this is not secure. We will see as in the case of JH a well designed padding rule leads to additional security guarantee.

Definition 4.2.3. A padding rule P is a tuple of two efficiently computable functions

$$P \equiv (\textsc{Pad} : \{0,1\}^* \to (\{0,1\}^n)^+, \textsc{Depad} : (\{0,1\}^n)^+ \to \{0,1\}^* \cup \{\perp\})$$

such that for any $M \in \{0,1\}^*$ we have

$$\textsc{Depad}(\textsc{Pad}(M)) = M.$$

$\textsc{Depad}(y)$ outputs $\perp$ if there exists no $M \in \{0,1\}^*$ such that, $\textsc{Pad}(M) = y$.

The function $\textsc{Pad}$ takes a message of arbitrary length and outputs the padded message which is multiple of block length. Where as, the function $\textsc{Depad}$ takes the padded message which is multiple of block length and outputs the original message. Normally, when we specify a padding rule we only specify the function PAD, but usually definition of DEPAD can be trivially derived from the description of PAD. In our context, we are interested in a specialized class of padding rules, namely with the following additional properties.

1. $\frac{|\text{PAD}(M)|}{n} = \lceil \frac{|M|}{n} \rceil + 1$.
2. For any $M \in (\{0,1\}^n)^+$, let $LB(M) \subseteq \{0,1\}^n$ be the set of n-bit elements (possible last blocks) such that for all $m \in \{0,1\}^n$,

   $$\text{DEPAD}(M\|m) \neq \perp .$$

   We want, $|LB(M)|$ to be small for all $M \in \{0,1\}^*$(smaller than some constant).

Here, if $x \in \{0,1\}^*$, $|x|$ denotes the length of x in bits. Also, if A is a set, $|A|$ denotes the number of elements in A. Any padding which satisfies the above two properties is called *good* padding rule. Now we are ready to define the JH mode of operation with padding.

Definition 4.2.4. With respect to a padding rule $P = (\text{PAD}, \text{DEPAD})$ and a permutation π, the JH_P mode of operation is defined as follows,

$$JH_P^\pi(\cdot) : \{0,1\}^* \to \{0,1\}^{2n-s} \equiv JH^\pi(\text{PAD}(\cdot)) \equiv \text{CHOPL}_s(MD^{f^\pi}(\text{PAD}(\cdot))).$$

The JH Padding rule:

In [113], the following padding rule is mentioned for JH hash function with block length $n = 512$. *Suppose that the length of the message M is $\ell(M)$ bits. Append the bit 1 to the end of the message, followed by $384 - 1 + (-\ell(M)$ mod $512)$ zero bits. Then the binary representation of $\ell(M)$ in big endian form is concatenated. This padding rule ensures that at least one block of 512 bits is padded after the message (irrespective of whether the message length is multiple of 512)* . It is easy to check the above padding rule is actually a *good* padding rule with $|LB(M)| \leq 2$.

4.3 Main Tools for Bounding Distinguisher's Advantage

We follow a similar approach to [32, 31] for proving indifferentiability security over here. We start with modeling the attacker. Then we construct a simulator, whose output distribution in attacker's view remains statistically close whether the attacker is interacting with JH Hash function and the random permutation it is based on, or is interacting with a random function and the simulator. Compared to [32] we do not restrict ourselves to some particular type of *irreducible views*. The underlying small domain oracle being a random permutation we also need to answer inverse queries.

Consistent Oracles

Intuitively, a small domain oracle is said to be consistent to a big domain oracle with respect to some mode of operation if querying the mode of operation based on the small domain oracle is equivalent to querying the big domain oracle.

Definition 4.3.1. A (small domain) probabilistic oracle algorithm G_2 is said to be *consistent* to a (big domain) probabilistic oracle algorithm G_1 with respect to MO-mode of operation if for any point x (from the big domain), we have

$$\Pr[G_1(x) = MO^{G_2}(x)] = 1.$$

Evaluatable queries

There might be some point x for which the value of $MO^{G_2}(x)$ gets fixed by the relations $G_2(x_1) = y_1, \cdots, G_2(x_q) = y_q$. Such x's are called evaluatable by the relations $G_2(x_1) = y_1, \cdots, G_2(x_q) = y_q$. Formally,

Definition 4.3.2. A point $x \in Domain(MO^{G_2})$ is called *evaluatable* with respect to MO-mode of operation (based on G_2) by the relations $G_2(x_1) = y_1, \cdots, G_2(x_q) = y_q$, if there exists a deterministic algorithm $\mathcal{B}$ such that,

$$\Pr[MO^{G_2}(x) = \mathcal{B}(x, (x_1, y_1), \cdots, (x_q, y_q)) | G_2(x_1) = y_1, \cdots, G_2(x_q) = y_q] = 1.$$

Modeling the adversary

In this chapter the adversary is modeled as a deterministic, computationally unbounded[1] distinguisher $\mathcal{A}$ which has access to two oracles $\mathcal{O}_1$ and $\mathcal{O}_2$. Recall that $\mathcal{A}$ tries to distinguish the output distribution of (JH^{π}, π) from that of (R, S^R). We say $\mathcal{A}$ queries $\mathcal{O}_1$ when it queries the oracle JH^{π} or R and queries $\mathcal{O}_2$ when it queries the oracle π or S^R. As we model π as a random permutation, the distinguisher is allowed to make inverse queries to oracle $\mathcal{O}_2$. We denote the forward query as $(\mathcal{O}_2(+, \cdot, \cdot))$ and inverse query as $(\mathcal{O}_2(-, \cdot, \cdot))$. The view $\mathcal{V}$ of the distinguisher is the list query-response tuple

$$((M_1, h_1), \ldots, (M_{q_1}, h_{q_1}), (x_1^1, x_1^2, y_1^1, y_1^2), \ldots, (x_{q_2+q_3}^1, x_{q_2+q_3}^2, y_{q_2+q_3}^1, y_{q_2+q_3}^2)) \tag{4.1}$$

Where,

$$\mathcal{O}_1(M_1) = h_1, \ldots, \mathcal{O}_1(M_{q_1}) = h_{q_1}$$
$$\mathcal{O}_2(+, x_1^1, x_1^2) = (y_1^1, y_1^2), \ldots, \mathcal{O}_2(+, x_{q_2}^1, x_{q_2}^2) = (y_{q_2}^1, y_{q_2}^2)$$
$$\mathcal{O}_2(-, y_{q_2+1}^1, y_{q_2+1}^2) = (x_{q_2+1}^1, x_{q_2+1}^2), \ldots, \mathcal{O}_2(-, y_{q_2+q_3}^1, y_{q_2+q_3}^2) = (x_{q_2+q_3}^1, x_{q_2+q_3}^2)$$

[1]Any deterministic adversary with unlimited resource is as powerful as a randomized adversary

Definition 4.3.3. For any view $\mathcal{V}$ as in (Equation 4.1), we define *Input View* $\mathcal{I}(\mathcal{V})$ and *Output View* $\mathcal{O}(\mathcal{V})$ as follows,

$$\mathcal{I}(\mathcal{V}) = (M_1, \ldots, M_q, (x_1^1, x_1^2), \ldots, (x_{q2}^1, x_{q2}^2), (y_{q2+1}^1, y_{q2+1}^2), \ldots, (y_{q2+q3}^1, y_{q2+q3}^2))$$
$$\mathcal{O}(\mathcal{V}) = (h_1, \ldots, h_q, (y_1^1, y_1^2), \ldots, (y_{q2}^1, y_{q2}^2), (x_{q2+1}^1, x_{q2+1}^2), \ldots, (x_{q2+q3}^1, x_{q2+q3}^2))$$

Below we point out some important observations,

1. $\mathcal{V}$, $\mathcal{I}(\mathcal{V})$ and $\mathcal{O}(\mathcal{V})$ are actually ordered tuples. That means, the position of any element inside the tuple actually denotes the corresponding query number. So, in general $\mathcal{O}_1(.)$, $\mathcal{O}_2(+, (.,.))$ and $\mathcal{O}_2(-, (.,.))$ queries should not be grouped together. But we write it like this to avoid further notational complexity.

2. For any deterministic *non-adaptive* attacker $\mathcal{I}(\mathcal{V})$ is always fixed.

3. For any deterministic *adaptive* attacker $\mathcal{I}(\mathcal{V})$ is actually determined by $\mathcal{O}(\mathcal{V})$ [89].

4. For any deterministic attacker (adaptive or non-adaptive) $\mathcal{V}$ is actually determined by $\mathcal{O}(\mathcal{V})$.

Irreducible Views

Loosely speaking an irreducible view does not contain any duplicate query, and none of the $\mathcal{O}_1$ queries are evaluatable from the $\mathcal{O}_2$ queries present in the view.

Definition 4.3.4. A view,

$$\mathcal{V} = ((M_1, h_1), \ldots, (M_{q_1}, h_{q_1}), (x_1^1, x_1^2, y_1^1, y_1^2), \ldots, (x_{q2+q3}^1, x_{q2+q3}^2, y_{q2+q3}^1, y_{q2+q3}^2))$$

is called *irreducible* if

- $M_1, \ldots, M_{q_1}$ are distinct,
- $(x_1^1, x_1^2), \ldots, (x_{q2+q3}^1, x_{q2+q3}^2)$ are distinct,
- (y_1^1, y_1^2), ..., $(y_{q2+q3}^1, y_{q2+q3}^2)$ are distinct,
- $M_1, \cdots, M_{q_1}$ are not evaluatable by the relations

  $$\pi(x_1^1, x_1^2) = (y_1^1, y_1^2), \ldots, \pi(x_{q2+q3}^1, x_{q2+q3}^2) = (y_{q2+q3}^1, y_{q2+q3}^2)$$

  with respect to MD-mode of operation based on f^π.

Also, any view which is not irreducible is called reducible view.

Definition 4.3.5. For an attacker $\mathcal{A}$, an output view $\mathcal{OV}$ is called *irreducible* if the corresponding view $\mathcal{V}$ is irreducible. Any output view which is not irreducible is called *reducible* output view.

Let $\mathcal{OV}^{\mathcal{A}}_{\mathcal{O}_1,\mathcal{O}_2}$ be the random variable corresponding to the output view of attacker $\mathcal{A}$, obtained after interacting with $\mathcal{O}_1, \mathcal{O}_2$. Also, $\mathcal{V}^{\mathcal{A}}_{\mathcal{O}_1,\mathcal{O}_2}$ be the random variable corresponding to the view of attacker $\mathcal{A}$, obtained after interacting with $\mathcal{O}_1, \mathcal{O}_2$.

The theorem below shows, if the probability distributions for all possible output views in two scenarios are close, then the attacker advantage is small. In contrast to the existing work in the literature, here we concentrate on *output views* instead of *views*. In fact, for a fixed attacker $\mathcal{A}$, there is always an one to one mapping between any view and output view.

Theorem 4.3.6. *F_i, G_i be the probabilistic oracle algorithms. If for an attacker $\mathcal{A}$, the relation*

$$\Pr[\mathcal{OV}^{\mathcal{A}}_{F_1,F_2} = \mathcal{OV}] \geq (1-\varepsilon)\Pr[\mathcal{OV}^{\mathcal{A}}_{G_1,G_2} = \mathcal{OV}],$$

holds for all possible output views $\mathcal{OV}$, then we have, $\mathsf{Adv}_{\mathcal{A}}((F_1,F_2),(G_1,G_2)) \leq \varepsilon$

In general it is hard to show the necessary condition of Theorem 4.3.6 for all possible output views. Theorem 4.3.7 proves that it is sufficient to work with irreducible output views instead of all possible output views. In fact, one can reduce any output view to an irreducible output view and then can apply Theorem 4.3.6.

Theorem 4.3.7. *If there exists a simulator S^R consistent to a random oracle R with respect to JH-mode of operation, such that for any attacker $\mathcal{A}$ making at most q queries, the relation*

$$\Pr[\mathcal{OV}^{\mathcal{A}}_{JH^\pi,\pi} = \mathcal{OV}] \geq (1-\varepsilon)\Pr[\mathcal{OV}^{\mathcal{A}}_{R,S^R} = \mathcal{OV}],$$

holds for all possible irreducible *output views $\mathcal{OV}$ (with respect to $\mathcal{A}$); then for any attacker $\mathcal{A}$ making at most q queries, we have*

$$\mathsf{Adv}_{\mathcal{A}}((JH^\pi,\pi),(R,S^R)) \leq \varepsilon.$$

Proof. This theorem differs from Theorem 4.3.6, only in the aspect that here probability distributions are close only for the irreducible output views. For any reducible output view $\mathcal{OV}$ and the corresponding attacker $\mathcal{A}$, let $\mathcal{V}$ be the view fixed by $\mathcal{OV}$ and $\mathcal{A}$. Let $\mathcal{V}'$ be the view obtained by deleting the computable $\mathcal{O}_1$ queries and repeated $\mathcal{O}_2$ queries of $\mathcal{V}$. The input view $\mathcal{I}(\mathcal{V}')$ actually specifies a non-adaptive attacker $\mathcal{A}'$. The output view $\mathcal{OV}' = \mathcal{O}(\mathcal{V}')$ is actually an irreducible output view with respect to $\mathcal{A}'$. As, π is consistent to JH^π and S^R is consistent to R with respect to JH-mode of operation we have,

$$\begin{aligned}\Pr[\mathcal{OV}^{\mathcal{A}}_{JH^\pi,\pi} = \mathcal{OV}] &= \Pr[\mathcal{OV}^{\mathcal{A}'}_{JH^\pi,\pi} = \mathcal{OV}']\\ \Pr[\mathcal{OV}^{\mathcal{A}}_{R,S^R} = \mathcal{OV}] &= \Pr[\mathcal{OV}^{\mathcal{A}'}_{R,S^R} = \mathcal{OV}'].\end{aligned}$$

Note, $\mathcal{A}'$ actually makes fewer queries compared to $\mathcal{A}$. Hence, even for reducible views, we have

$$\begin{aligned}\Pr[\mathcal{OV}^{\mathcal{A}}_{JH^\pi,\pi} = \mathcal{OV}] &= \Pr[\mathcal{OV}^{\mathcal{A}'}_{JH^\pi,\pi} = \mathcal{OV}']\\ &\geq (1-\varepsilon)\Pr[\mathcal{OV}^{\mathcal{A}'}_{R,S^R} = \mathcal{OV}']\\ &= (1-\varepsilon)\Pr[\mathcal{OV}^{\mathcal{A}}_{R,S^R} = \mathcal{OV}].\end{aligned}$$

So the required condition of Theorem 4.3.6 remains true. Now, by applying Theorem 4.3.6 we get the result.

□

4.4 Indifferentiability Security Analysis of JH'

4.4.1 Simulator and its Interpolation Probability

The simulator maintains one partial permutation $e_1 : \{0,1\}^{2n} \to \{0,1\}^{2n}$ initially empty, one partial function $e_1^* : (\{0,1\}^n)^* \to \{0,1\}^{2n}$ initialized with $e_1^*(\phi) = IV_1 \| IV_2$. It also maintains two sets C_1, C_2 initialized as $C_1 = \{IV_1\}$ and C_2 as empty. Let I_1 denotes the set of points on which e_1 is defined, O_1 denotes the output points of e_1. $FH, LH : \{0,1\}^{2n} \to \{0,1\}^n$ be the two functions outputting first n-bits and last n-bits of any $2n$-bit number respectively.

The goal of the simulator is to remain consistent with R with respect to JH'-mode of operation while behaving like a random permutation. Before describing the simulator, we give some insight informally on how the simulator works.

1. In the partial permutation e_1, the simulator maintains its history.
2. In the partial function e_1^*, the simulator maintains the list of queries evaluatable by e_1 with respect to JH-mode of operation.
3. C_1 is the set of *first half* (first n-bits) of e_1^* outputs.
4. Even though e_1^* is evaluatable by the partial permutation e_1, it might happen that e_1 is also defined at some points which do not help in evaluating e_1^*. C_2 is the set of first half of such points.
5. The simulator makes sure, C_1 and C_2 always remain mutually exclusive.
6. Because of 5, there are no so called *accidents*. That means when the attacker is interacting with (R, S^R) and she wants to evaluate $\mathcal{O}_1(m_1 \| \cdots \| m_\ell)$ through a series of $\mathcal{O}_2$ queries, she will always have to make a series of ℓ queries starting with $\mathcal{O}_2(IV_1, IV_2 \oplus m_1)$. The attacker cannot hope to skip a query in the middle.

We note at any point of time, the following conditions hold.

$$|O_1| \le q_2 + q_3 \textit{ and } |I_1| \le q_2 + q_3 \textit{ and } |C_1 \cup C_2| \le q_2 + q_3 \textit{ and } |C_1| \le q_2 + 1$$

Theorem 4.4.1. *For any attacker $\mathcal{A}$ against JH' and any irreducible output view $\mathcal{OV}$ with respect to it, we have*

$$\Pr[\mathcal{OV}^{\mathcal{A}}_{R,S'^R} = \mathcal{OV}] \le \frac{1}{2^{(2n-s)q_1 + 2n(q_2+q_3)}} \times \frac{1}{(1 - \frac{2(q_2+q_3)}{2^{\min(s,n)}})^{q_2}} \times \frac{1}{(1 - \frac{2(q_2+q_3)}{2^n})^{q_3}}$$

```
$S'^R(+, x_1, x_2)$
• IF $e_1(x_1\|x_2) = z$ RETURN $z$
• IF there exists $M$, s.t $e_1^*(M) = x_1\|x'$
    1. $m = x' \oplus x_2$
    2. $y = R(M\|m) \oplus \text{CHOPL}(m\|0^n)$
    3. $w \in_R \{0,1\}^s$
    4. $z = w\|y$
    5. IF ( $z \in O_1$ OR $FH(z) \oplus m \in C_1 \cup C_2$)
        – GOTO Step 3
    6. $C_1 = C_1 \cup \{FH(z) \oplus m\}$
    7. $e_1^*(M\|m) = z \oplus (m\|0^n)$
    8. $e_1(x_1\|x_2) = z$
    9. RETURN $z$
• ELSE
    10. $z \in_R \{0,1\}^{2n}$
    11. IF $z \in O_1$
        – GOTO Step 10
    12. $e_1(x_1\|x_2) = z$
    13. $C_2 = C_2 \cup \{x_1\}$
    14. RETURN $z$
```

```
$S'^R(-, y_1, y_2)$
• IF there exists $z_1\|z_2$ such that $e_1(z_1\|z_2) = y_1\|y_2$
    – RETURN $z_1\|z_2$
• ELSE
    1. $z_1 \in_R \{0,1\}^n$
    2. IF $z_1 \in C_1$
        – GOTO Step 1
    3. $z_2 \in_R \{0,1\}^n$
    4. IF $z_1\|z_2 \in I_1$
        – GOTO Step 3
    5. $C_2 = C_2 \cup \{z_1\}$
    6. RETURN $z_1\|z_2$
```

Figure 4.4: Simulator for JH'

where $2(q_2+q_3) < 2^{min(s,n)}$.

Proof. As $\mathcal{OV}$ is irreducible, R query outputs are independent of the other queries, hence R being a Random Function for q_1 many R queries we get the term $\frac{1}{2^{(2n-s)q_1}}$. For an $S'^R(+,\cdot,\cdot)$ queries, simulator is giving output as $w\|y$, there are two scenarios.

1. y is distributed uniformly over $\{0,1\}^{2n-s}$ and w is distributed uniformly over $\{0,1\}^s \setminus \{z \in \{0,1\}^s : z\|y \in O_1 \text{ or } FH(z\|y) \oplus (x' \oplus x_2) \in C_1 \cup C_2\}$.

2. $w\|y$ is distributed uniformly over $\{0,1\}^{2n} \setminus O_1$.

By Lemma 4.2.2 we know,

$$|\{z \in \{0,1\}^s : y\|z \in O_1\}| \le |O_1| \le (q_2+q_3).$$

On the other hand, using Lemma 4.2.2 here we have,

$$\begin{aligned}|\{z \in \{0,1\}^s : FH(z\|y) \oplus (x' \oplus x_2) \in C_1 \cup C_2\}| &\le \frac{2^s}{2^{\min(s,n)}}|C_1 \cup C_2| \\ &\le \frac{2^s}{2^{\min(s,n)}}(q_2+q_3).\end{aligned}$$

Hence, for any $(w\|y) \in \{0,1\}^{2n}$ we have,

$$\begin{aligned}&\Pr[S'^R(+,\cdot,\cdot) \text{ query outputs } (w\|y)] \\ &\le \max\left(\frac{1}{2^{2n-s}}\frac{1}{2^s - \frac{2^s}{2^{\min(s,n)}}(q_2+q_3) - (q_2+q_3)}, \frac{1}{2^{2n}-(q_2+q_3)}\right) \\ &\le \frac{1}{2^{2n}}\frac{1}{(1-\frac{2(q_2+q_3)}{2^{\min(s,n)}})}\end{aligned}$$

For $S^R(-,\cdot,\cdot)$ query giving output as $z_1\|z_2$ we know,

1. z_1 is uniformly distributed over $\{0,1\}^n \setminus C_1$

2. z_2 is uniformly distributed over $\{0,1\}^n \setminus \{w \in \{0,1\}^n : z_1\|w \in I_1\}$

We know, $|C_1| \le (q_2+1)$ and $|I_1| \le (q_2+q_3)$. Hence, for any $(z_1\|z_2) \in \{0,1\}^{2n}$ we have

$$\begin{aligned}\Pr[S^R(-,\cdot,\cdot) \text{ query outputs } (z_1\|z_2)] &\le \frac{1}{(2^n-(q_2+1))(2^n-(q_2+q_3))} \\ &\le \frac{1}{2^{2n}}\frac{1}{1-\frac{2(q_2+q_3)}{2^n}}\end{aligned}$$

Hence, all together we get

$$\Pr[\mathcal{OV}^{\mathcal{A}}_{R,S^R} = \mathcal{OV}] \le \frac{1}{2^{(2n-s)q_1+2n(q_2+q_3)}} \times \frac{1}{(1-\frac{2(q_2+q_3)}{2^{\min(s,n)}})^{q_2}} \times \frac{1}{(1-\frac{2(q_2+q_3)}{2^n})^{q_3}}$$

□

Next we wish to show that our simulator is efficient. We know $|O_1| \leq (q_2 + q_3)$ and $|C_1 \cup C_2| \leq (q_2 + q_3)$. The probability that the GOTO statement at Step 5 in forward query in Figure 4.4 gets executed is at most $\frac{2(q_2+q_3)}{2^{\min(s,n)}}$. The condition $2^{\min(s,n)} > 4(q_2 + q_3)$ ensures that it happens with probability at most $\frac{1}{2}$ in each iteration. Hence except with negligible probability, Step 5 takes at most $\mathcal{O}(q_2 + q_3)$ time to satisfy the condition. The same argument holds for other GOTO statements as well. Hence we get the following result.

Theorem 4.4.2. *If* $2^{\min(s,n)} > 4(q_2 + q_3)$*, the simulator* S'^R *takes at most* $\mathcal{O}(q_2 + q_3)$ *time to answer any query (except with exponentially small probability).*

4.4.2 Interpolation Probability of $\mathcal{OV}^{\mathcal{A}}_{JH'^{\pi},\pi}$

In Theorem 4.4.1 we have shown upper bound for $\Pr[\mathcal{OV}^{\mathcal{A}}_{R,S'^R} = \mathcal{OV}]$ for any irreducible output views $\mathcal{OV}$. The Theorem below gives a lower bound for $\Pr[\mathcal{OV}^{\mathcal{A}}_{JH'^{\pi},\pi} = \mathcal{OV}]$ for any irreducible output view $\mathcal{OV}$. Later we will apply Theorem 4.3.7 to prove the indifferentiability bound using these upper and lower bounds.

Theorem 4.4.3. *For any attacker* $\mathcal{A}$ *and any irreducible output view* $\mathcal{OV}$ *with respect to it, we have*

$$\Pr[\mathcal{OV}^{\mathcal{A}}_{JH'^{\pi},\pi} = \mathcal{OV}] \geq \frac{1}{2^{(2n-s)q_1+2n(q_2+q_3)}} \times (1 - \frac{2\sigma^2}{2^{2n}}) \times (1 - \frac{2q_1(q_1+q_2+q_3)}{2^{\min(s,n)}}).$$

The proof of the above theorem involves two steps. Starting with an attacker $\mathcal{A}$ against $JH'^{\pi} \equiv \text{CHOPL}_s(MD^{f^{\pi}})$ we construct another attacker $\mathcal{A}'$ against $MD^{f^{\pi}}$ which essentially makes same queries as $\mathcal{A}$ but has access to unchopped output view.

- First we define the notion of *MD-irreducible* view (irreducible view with respect to Merkle-Damgård mode of operation) and then we show for the output view $\mathcal{OV}_{MD}$ corresponding to any *MD-irreducible* view we actually have,

$$\Pr[\mathcal{OV}^{\mathcal{A}'}_{MD^{f^{\pi}},\pi} = \mathcal{OV}_{MD}] \geq \frac{1}{2^{2nq_1+2n(q_2+q_3)}} \times (1 - \frac{2\sigma^2}{2^{2n}})$$

- In Theorem 4.4.7 we show, given an irreducible output view $\mathcal{OV}$ and an attacker $\mathcal{A}$, if $\mathbf{OV}_{MD}$ is the set of all *MD-irreducible* output views for the attacker $\mathcal{A}'$ such that,

$$\Pr[\mathcal{OV}^{\mathcal{A}}_{JH'^{\pi},\pi} = \mathcal{OV} | \mathcal{OV}^{\mathcal{A}'}_{MD^{f^{\pi}},\pi} = \mathcal{OV}_{MD}] = 1$$

for all $\mathcal{OV}_{MD} \in \mathbf{OV}_{MD}$; then

$$|\mathbf{OV}_{MD}| \geq 2^{sq_1} \times (1 - \frac{2q_1(q_1+q_2+q_3)}{2^{\min(s,n)}})$$

The above two results will readily imply Theorem 4.4.3.

Definition 4.4.4. The set of relations

$$MD^{f^{\mathcal{O}_2}}(M_1\|m_1) = g_1, \ldots, MD^{f^{\mathcal{O}_2}}(M_{q_1}\|m_{q_1}) = g_{q_1}$$
$$\mathcal{O}_2(x_1^1, x_1^2) = (y_1^1, y_1^2), \ldots, \mathcal{O}_2(x_{q_2+q_3}^1, x_{q_2+q_3}^2) = (y_{q_2+q_3}^1, y_{q_2+q_3}^2)$$

is *MD-irreducible* if,

1. $g_1 \oplus (m_1\|0^n), \ldots, g_{q_1} \oplus (m_{q_1}\|0^n), y_1^1\|y_1^2, \ldots, y_{q_2+q_3}^1\|y_{q_2+q_3}^2$ are all different.
2. For $i = 1, \ldots, q_1$, one of the following two conditions holds
   (a) $FH(g_i)$ is different from $x_1^1, \ldots, x_{q_2+q_3}^1$ and IV_1.
   (b) Σ be the set of all message blocks present in $MD^{f^{\mathcal{O}_2}}$ queries. If $FH(g_i) = IV_1$, then $LH(g_i) \oplus IV_2 \notin \Sigma$. If $FH(g_i) = x_j^1$ for some $1 \le j \le q_2 + q_3$, then $LH(g_i) \oplus x_j^2 \notin \Sigma$.
3. $M_1\|m_1, \ldots, M_{q_1}\|m_{q_1}$ are not evaluatable by the relations

$$\mathcal{O}_2(x_1^1, x_1^2) = (y_1^1, y_1^2), \ldots, \mathcal{O}_2(x_{q_2+q_3}^1, x_{q_2+q_3}^2) = (y_{q_2+q_3}^1, y_{q_2+q_3}^2)$$

   with respect to MD-mode of operations based on $f^{\mathcal{O}_2}$

We also say the tuple,

$$v = ((M_1\|m_1, g_1), \ldots, (M_{q_1}\|m_{q_1}, g_{q_1}), (x_1^1, x_1^2, y_1^1, y_1^2), \ldots,$$
$$(x_{q_2+q_3}^1, x_{q_2+q_3}^2, y_{q_2+q_3}^1, y_{q_2+q_3}^2))$$

is *MD-irreducible* if and only if the corresponding set of relations is MD-irreducible, according to Definition 4.4.4.

The definition above is similar to the definition of irreducible view (Definition 4.3.4). But here we are interested in the view without any chopping. Note, condition 2 ensures $M_i\|m_i$ is not evaluatable even with the help of the relations $MD^{f^{\mathcal{O}_2}}(M_j\|m_j) = h_j$ for $j \neq i$. Loosely speaking, the Theorem below gives a lower bound of the probability of getting a particular MD-irreducible tuple v, when a attacker interacts with (MD^{f^π}, π).

Theorem 4.4.5. *Let a tuple*

$$v = ((M_1\|m_1, g_1), \ldots, (M_{q_1}\|m_{q_1}, g_{q_1}), (x_1^1, x_1^2, y_1^1, y_1^2), \ldots,$$
$$(x_{q_2+q_3}^1, x_{q_2+q_3}^2, y_{q_2+q_3}^1, y_{q_2+q_3}^2))$$

be MD-irreducible*, then the number of permutations π such that,*

$$MD_{IV_1\|IV_2}^{f^\pi}(M_1\|m_1) = g_1, \ldots, MD_{IV_1\|IV_2}^{f^\pi}(M_{q_1}\|m_{q_1}) = g_{q_1}$$
$$\pi(x_1^1, x_1^2) = (y_1^1, y_1^2), \ldots, \pi(x_{q_2+q_3}^1, x_{q_2+q_3}^2) = (y_{q_2+q_3}^1, y_{q_2+q_3}^2) \ldots \textit{Rel B}$$

is at least

$$\frac{|\Pi|}{2^{2nq_1+2n(q_2+q_3)}} \times (1 - \frac{2\sigma^2}{2^{2n}}),$$

where $|\Pi| = (2^{2n})!$ *is the total number of permutations from* $\{0,1\}^{2n}$ *to* $\{0,1\}^{2n}$ *and* σ *is the total number of message blocks queried. Also for a MD-irreducible tuple* v*, the probability that Rel B holds is at least*

$$\frac{1}{2^{2nq_1+2n(q_2+q_3)}} \times (1 - \frac{2\sigma^2}{2^{2n}}),$$

when π *is a random permutation.*

Proof. Let D be the set of all elements from $(\{0,1\}^n)^+$ whose $MD^{f^\pi}_{IV_1\|IV_2}$ values are determined from the relations

$$\pi(x^1_1, x^2_1) = (y^1_1, y^2_1), \ldots, \pi(x^1_{q_2+q_3}, x^2_{q_2+q_3}) = (y^1_{q_2+q_3}, y^2_{q_2+q_3}).$$

Since v is MD-irreducible, $M_i\|m_i \notin D$ for all $1 \le i \le q_1$. let P denote the set of all nonempty prefixes of M_i's. More precisely,

$$P = \{M \in (\{0,1\}^n)^+ : M \textit{ is prefix of } M_i \textit{ for some } 1 \le i \le q_1\}.$$

We enumerate the set $P \setminus D \equiv \{N_1, \ldots, N_{\sigma'}\}$. Note that, $|P| + q_1 \le \sum_i \|M_i\|$. Now, we have

$$\sigma = q_2 + q_3 + \sum_i \|M_i\| \ge q_2 + q_3 + |P| + q_1 \ge q_1 + q_2 + q_3 + \sigma' \equiv \sigma''$$

Note that, $\sigma' = |P \setminus D| \le |P|$. As the number of feasible choices depend only on σ' and not on $|P|$, we used σ' in the previous expression.

We can choose outputs of $MD^{f^{\mathcal{O}_2}}_{IV_1\|IV_2}(N_1), \ldots, MD^{f^{\mathcal{O}_2}}_{IV_1\|IV_2}(N_{\sigma'})$ in at least

$$(2^{2n} - 2(q_1 + q_2 + q_3))(2^{2n} - 2(q_1 + q_2 + q_3 + 1)) \ldots (2^{2n} - 2(q_1 + q_2 + q_3 + \sigma' - 1))$$

ways. (In the negative term, the factor 2 comes because, any output value should not be same as other output values and the next input value induced by the output value should not be same as other input values.) Hence,

$$\begin{aligned}
&|\{\pi : \{0,1\}^{2n} \to \{0,1\}^{2n} \textit{ such that } \pi \textit{ is a permutation and satisfies Rel B}\}| \\
&\ge (2^{2n} - \sigma'')! \times \prod_{i=0}^{\sigma'-1} (2^{2n} - 2(q_1 + q_2 + q_3 + i)) \\
&\ge \frac{(2^{2n})!}{2^{2n\sigma''}} \times 2^{2n\sigma'} \times (1 - \frac{2\sigma'\sigma''}{2^{2n}}) \ge \frac{|\Pi|}{2^{2nq_1+2n(q_2+q_3)}} \times (1 - \frac{2\sigma^2}{2^{2n}})
\end{aligned}$$

For the third inquality we used $\sigma > \sigma'' > \sigma'$ and $\sigma'' = q_1 + q_2 + q_3 + \sigma'$. Π is the set of all permutations over $\{0,1\}^{2n}$.

□

Definition 4.4.6. With respect to an irreducible view

$$\mathcal{V} = ((M_1\|m_1, h_1), \ldots, (M_{q_1}\|m_{q_1}, h_{q_1}), (x_1^1, x_1^2, y_1^1, y_1^2), \ldots,$$

$$(x_{q_2+q_3}^1, x_{q_2+q_3}^2, y_{q_2+q_3}^1, y_{q_2+q_3}^2))$$

an MD-irreducible tuple v is said to be CHOPL_s*-matching* if

$$v = ((M_1\|m_1, w_1\|h_1), \ldots, (M_{q_1}\|m_{q_1}, w_{q_1}\|h_{q_1}), (x_1^1, x_1^2, y_1^1, y_1^2), \ldots,$$

$$(x_{q_2+q_3}^1, x_{q_2+q_3}^2, y_{q_2+q_3}^1, y_{q_2+q_3}^2)),$$

for some q_1-tuple $w = (w_1, \ldots, w_{q_1})$.

Let $M'_{\mathcal{V}}$ be the set of all such CHOPL_s*-matching* MD-irreducible tuples.

Theorem 4.4.7. *For any irreducible view*

$$\mathcal{V} = ((M_1\|m_1, h_1), \ldots, (M_{q_1}\|m_{q_1}, h_{q_1}), (x_1^1, x_1^2, y_1^1, y_1^2), \ldots,$$

$$(x_{q_2+q_3}^1, x_{q_2+q_3}^2, y_{q_2+q_3}^1, y_{q_2+q_3}^2))$$

we have,

$$|M'_{\mathcal{V}}| \geq 2^{sq_1} \times (1 - \frac{q_1(q_1 + q_2 + q_3)}{2^{\min(s,n)}}).$$

Proof. By Lemma 4.2.2, we know

$$\begin{aligned}&|\{z \in \{0,1\}^s : z\|h_1 \oplus (m_1\|0^n) \in \{y_1^1\|y_1^2, \ldots, y_{q_2+q_3}^1\|y_{q_2+q_3}^2\}\}| \\ &= |\{z \in \{0,1\}^s : z\|h_1 \in \{y_1^1\|y_1^2 \oplus (m_1\|0^n), \ldots, y_{q_2+q_3}^1\|y_{q_2+q_3}^2 \oplus (m_1\|0^n)\}\}| \leq q_2 + q_3\end{aligned}$$

and

$$|\{z \in \{0,1\}^s : FH(z\|h_1) \in \{x_1^1, \ldots, x_{q_2+q_3}^1, IV_1\}\}| \leq \frac{2^s}{2^{\min(s,n)}}(q_2 + q_3 + 1).$$

Hence there are at least $(2^s - (q_2 + q_3 + \frac{2^s}{2^{\min(s,n)}}(q_2 + q_3 + 1)))$ many possible values for w_1. Similarly once w_1 is selected there are at least $(2^s - (q_2 + q_3 + \frac{2^s}{2^{\min(s,n)}}(q_2 + q_3 + 1) + 1))$ many possible values for w_2 and so on. Hence,

$$\begin{aligned}|M'_{\mathcal{V}}| = &\text{ Number of valid } w \text{ tuples} \\ &\geq (2^s - (q_2 + q_3 + \frac{2^s}{2^{\min(s,n)}}(q_2 + q_3 + 1))) \ldots (2^s - (q_2 + q_3 + \frac{2^s}{2^{\min(s,n)}}(q_2 + q_3 + 1) + q_1 - 1)) \\ &\geq 2^{sq_1} \times (1 - \frac{2q_1(q_1 + q_2 + q_3)}{2^{\min(s,n)}})\end{aligned}$$

□

Now we are ready to prove Theorem 4.4.3, with help of Theorem 4.4.5 and Theorem 4.4.7. Let $\mathcal{V}$ be the irreducible view determined by $\mathcal{A}$ and irreducible output view $\mathcal{OV}$. Consider an Attacker $\mathcal{A}'$, which makes queries at the same input points as of $\mathcal{A}$, but has access to $MD^{f^{\mathcal{O}_2}}$ instead of $JH'^{\mathcal{O}_2}$. Hence,

$$\begin{aligned}
&\Pr[\mathcal{OV}^{\mathcal{A}}_{JH'^{\pi},\pi} = \mathcal{OV}] = \Pr[\mathcal{V}^{\mathcal{A}}_{JH'^{\pi},\pi} = \mathcal{V}] = \sum_{v \in M_{\mathcal{V}}} \Pr[\mathcal{V}^{\mathcal{A}'}_{MD^{f^{\pi}},\pi} = v] \\
&\geq \sum_{v \in M_{\mathcal{V}}} \frac{1}{2^{2nq_1 + 2n(q_2+q_3)}} \times (1 - \frac{2\sigma^2}{2^{2n}}) \\
&\geq \frac{1}{2^{2nq_1+2n(q_2+q_3)}} \times (1 - \frac{2\sigma^2}{2^{2n}}) \times 2^{sq_1} \times (1 - \frac{2q_1(q_1+q_2+q_3)}{2^{\min(s,n)}}) \\
&= \frac{1}{2^{(2n-s)q_1+2n(q_2+q_3)}} \times (1 - \frac{2\sigma^2}{2^{2n}}) \times (1 - \frac{2q_1(q_1+q_2+q_3)}{2^{\min(s,n)}})
\end{aligned}$$

4.4.3 Indifferentiability Security Bound

We are now ready to prove the main result of this section. For any attacker $\mathcal{A}$, making at most q_1, q_2, q_3 queries to the oracles $\mathcal{O}_1, \mathcal{O}_2(+,\cdot,\cdot), \mathcal{O}_2(-,\cdot,\cdot)$ respectively we show an upper bound for $\mathsf{Adv}_{\mathcal{A}}$.

Theorem 4.4.8. *The JH'^{π}-construction (with $(2n-s)$-bit output) based on a random permutation π is $(\mathcal{O}(q_2+q_3)), q_1, q_2+q_3, \epsilon)$ indifferentiable from a random oracle R, with*

$$\epsilon \leq \frac{2\sigma^2}{2^{2n}} + \frac{2q_3(q_2+q_3)}{2^n} + \frac{2q_2(q_2+q_3) + 2q_1(q_1+q_2+q_3)}{2^{\min(s,n)}},$$

where σ is the maximum number of message blocks queried, q_1 is the maximum number queries to JH'^{π} or R, q_2+q_3 is the maximum number of queries to π, π^{-1} or $S'^{R}(+,\cdot,\cdot), S^{R}(-,\cdot,\cdot)$. Here we also assume, $q_2+q_3 < 2^{\min(s,n)-2}$[1].

Proof. For any attacker $\mathcal{A}$ and an irreducible output view $\mathcal{OV}$ from Theorem 4.4.1 and Theorem 4.4.3 we have,

$$\begin{aligned}
&\Pr[\mathcal{OV}^{\mathcal{A}}_{JH^{\pi},\pi} = \mathcal{OV}] \\
&\geq \left(1 - \left(\frac{2\sigma^2}{2^{2n}} + \frac{2q_3(q_2+q_3)}{2^n} + \frac{2q_2(q_2+q_3) + 2q_1(q_1+q_2+q_3)}{2^{\min(s,n)}}\right)\right) \\
&\quad \times \Pr[\mathcal{OV}^{\mathcal{A}}_{R,S'^{R}} = \mathcal{OV}]
\end{aligned}$$

Now, applying Theorem 4.3.7 we get the required result. □

When maximum query length ℓ is smaller than $2^{n/2}$, for any attacker $\mathcal{A}$ (making at most q many queries) against the JH' construction we have

$$\mathsf{Adv}_{\mathcal{A}} = \mathcal{O}(\frac{q^2}{2^{\min(s,n)}})$$

[1]Note that, as q_2, q_3 are the query complexity of the adversary (rather than time complexity), the condition $q_2 + q_3 < 2^{\min(s,n)-2}$ does not restrict the adversary to be PPTM.

4.5 Indifferentiability Security Analysis of JH_P

4.5.1 Simulator and its Interpolation Probability

We describe our simulator in Fig 4.5. Similar to previous section, the following notation we used in describing the simulator.

- Partial permutation $e : \{0,1\}^{2n} \to \{0,1\}^{2n}$, initially empty. I denotes set of points where e is defined and O denotes the output points of e.
- Partial function $e^* : (\{0,1\}^n)^* \to \{0,1\}^{2n}$ initialized to $e^*(\phi) = IV_1 \| IV_2$.
- Set $C \subseteq \{0,1\}^n$ initialized to $C = \{IV_1\}$ is the FH (first half) of e^* outputs.

For a padding rule $P = (\text{PAD}, \text{DEPAD})$ and $M \in (\{0,1\}^n)^+$, we recall $LB(M) \subseteq \{0,1\}^n$ is defined as $\{m \in \{0,1\}^n : \text{DEPAD}(M\|m) \neq \perp\}$. As in case of the actual JH padding rule we assume, $|LB(M)| \leq 2$.

We recall the design philosophy behind the JH' simulator from Section 4.4.1. Over there the simulator was maintaining a list of *evaluatable* queries and their non-chopped outputs in the partial permutation e_1^*. When the simulator receives some query the goal of the simulators are three fold.

1. Give a random output keeping in mind the permutation property.
2. Do not create some new evaluatable query unless forced to do so. That means output of the simulator will never create a new evaluatable query with the exception of the following scenario.
3. It might happen, only the input of the simulator forces another new evaluatable query. (This happens if attacker is trying to find some $\mathcal{O}_1$ query output through $\mathcal{O}_2$ query.) If this happens, then adjust the output of the simulator so that it remains *consistent* to R, w.r.t. the new evaluatable query.

One crucial point is, during one simulator query the simulator must prevent creation of more than one evaluatable query. Otherwise, the simulator cannot remain consistent to both of them. In forward queries to JH' simulator with $s = n$, when the attacker has forced creation of one new evaluatable query the LH (last half) of the possible output gets fixed by R response of that evaluatable query, but the simulator has control over FH output with which it makes sure, another evaluatable query is not created.

Here the situation is reversed. FH gets fixed by R, the simulator has control only over LH. This is problematic, because only FH can lead to creation of more evaluatable queries (with one more message block after the current evaluatable query). In fact, in Section 4.6 the attacker against JH mode operation (without length padding at last block) exploits this fact. But the simulator can play with LH to change the actual evaluatable query (even though it cannot prevent the creation.) By doing so, the simulator ensures the new evaluatable query is not a valid padded message, hence for that query the simulator does not need to be consistent with R. The simulator also needs to be careful such that no new evaluatable queries of length (current evaluatable query length + 2) or more are created. However, that can easily be handled.

The next two theorems describe the running time and interpolation probability upper bound corresponding to the simulator.

Theorem 4.5.1. *For any attacker $\mathcal{A}$ against JH_P^{π} mode of operation and any irreducible output view $\mathcal{OV}$ with respect to it, we have*

$$\Pr[\mathcal{OV}^{\mathcal{A}}_{R,S^R} = \mathcal{OV}] \leq \frac{1}{2^{(2n-s)q_1+2n(q_2+q_3)}} \times \frac{1}{(1-\frac{(q_2+q_3+3)^2}{2^s})^{q_2}} \times \frac{1}{(1-\frac{(q_2+q_3+1)^2}{2^n})^{q_3}}$$

when $(q_2+q_3+3)^2 < 2^{\min(s,n)}$.

Proof. As $\mathcal{OV}$ is irreducible, the distribution R query outputs is independent from the output distribution of other queries, hence R being a Random Function for q_1 many R queries we get the term $\frac{1}{2^{(2n-s)q_1}}$. For $S^R(+,\cdot,\cdot)$ query giving output as $y\|w$ we actually have two scenarios:

1. y is distributed uniformly over $\{0,1\}^{2n-s}$ and w is distributed uniformly over $\{0,1\}^s \setminus (B_1 \cup B_2 \cup \cdots \cup B_7)$, where
   (a) $B_1 = \{z \in \{0,1\}^s : y\|z \in O\}$
   (b) $B_2 = \{z \in \{0,1\}^s : FH(y\|z) \oplus x_1 = m, LH(y\|z) \oplus x_2 \in LB(M\|m)\}$
   (c) $B_3 = \{z \in \{0,1\}^s : FH(y\|z) \oplus i_1 = m, LH(y\|z) \oplus i_2 \in LB(M\|m)$ for some $i_1\|i_2 \in I\}$
   (d) $B_4 = \{z \in \{0,1\}^s : FH(y\|z) \oplus x_1 = m, LH(y\|z) \oplus x_2 = FH(y\|z) \oplus x_1\}$
   (e) $B_5 = \{z \in \{0,1\}^s : FH(y\|z) \oplus x_1 = m, LH(y\|z) \oplus x_2 = FH(y\|z) \oplus i_1'$ for some $i_1'\|i_2' \in I\}$
   (f) $B_6 = \{z \in \{0,1\}^s : FH(y\|z) \oplus i_1 = m, LH(y\|z) \oplus i_2 = FH(e(i_1\|i_2)) \oplus x_1$ for some $i_1\|i_2 \in I\}$
   (g) $B_7 = \{z \in \{0,1\}^s : FH(y\|z) \oplus i_1 = m, LH(y\|z) \oplus i_2 = FH(e(i_1\|i_2)) \oplus i_1'$for some$i_1\|i_2, i_1'\|i_2' \in I\}$
2. $y\|w$ is uniformly distributed over $\{0,1\}^{2n} \setminus O$

```
$S^R(+, x_1, x_2)$

1. IF $e(x_1\|x_2) = z$ RETURN z
2. IF there exists $M$, s.t. $e^*(M) = x_1\|x'$
   (a) $m = x' \oplus x_2$
   (b) IF $M \neq \phi$ AND $m \in LB(M)$
       i. $y = R(\text{DEPAD}(M\|m)) \oplus \text{CHOPR}(m\|0^n)$
       ii. $w \in_R \{0,1\}^s$
       iii. $z = y\|w$
   (c) ELSE
       i. $z \in_R \{0,1\}^{2n}$
   (d) IF $z \in O$ GOTO Step 2b
   (e) $e^{*\prime} = e^*$
   (f) $C' = C$
   (g) FOR EACH $i_1\|i_2 \in I \cup \{x_1\|x_2\}$
       i. IF $FH(z) \oplus m \neq i_1$ CONTINUE
       ii. IF $LH(z) \oplus i_2 \in LB(M\|m)$
           • GOTO Step 2b
       iii. IF $i_1\|i_2 = x_1\|x_2$
           • $o_1\|o_2 = z$
       iv. ELSE
           • $o_1\|o_2 = e(i_1\|i_2)$
       v. $e^{*\prime}(M\|m\|LH(z) \oplus i_2) = (o_1 \oplus LH(z) \oplus i_2)\|o_2$
       vi. $C' = C' \cup \{o_1 \oplus LH(z) \oplus i_2\}$
       vii. FOR EACH $i_1'\|i_2' \in I \cup \{x_1\|x_2\}$
           • IF $LH(z) \oplus i_2 = o_1 \oplus i_1'$
             – GOTO Step 2b
   (h) $e^* = e^{*\prime}$
   (i) $C = C'$
   (j) $e^*(M\|m) = z \oplus (m\|0^n)$
   (k) $C = C \cup \{FH(z) \oplus m\}$
3. ELSE
   (a) $z \in_R \{0,1\}^{2n}$
   (b) IF $z \in O$ GOTO Step 3a
4. $e(x_1\|x_2) = z$
5. RETURN $z$
```

```
$S^R(-, y_1, y_2)$

1. IF there exists $z_1\|z_2$ such that $e(z_1\|z_2) = y_1\|y_2$
   • RETURN $z_1\|z_2$
2. ELSE
   (a) $z_1 \in_R \{0,1\}^n$
   (b) IF $z_1 \in C$
       • GOTO Step 2a
   (c) $z_2 \in_R \{0,1\}^n$
   (d) IF $z_1\|z_2 \in I$
       • GOTO Step 2c
   (e) $e(z_1\|z_2) = y_1\|y_2$
   (f) RETURN $z_1\|z_2$
```

Figure 4.5: Simulator for JH with padding

We know $|O| = |I| \leq q_2 + q_3$, we would also like to have upper bounds for $|B_1|, |B_2|, \cdots, |B_7|$.

1. By Lemma 4.2.2, $|B_1| \leq |O| \leq q_2 + q_3$. Also, $|B_2| \leq |LB(M\|m)| \leq 2$
2. We partition I as $I_1 \cup I_2 \cup \cdots$ depending on the *first half* values, more precisely $a, b \in I_j$ implies $FH(a) = FH(b)$. Now,

$$\begin{aligned}|B_3| &= |\{z \in \{0,1\}^s : FH(y\|z) \oplus i_1 = m, LH(y\|z) \oplus i_2 \in LB(M\|m) \text{ for some} \\ &\qquad i_1\|i_2 \in I\}| \\ &= \sum_j |\{z \in \{0,1\}^s : FH(y\|z) \oplus i_1 = m, LH(y\|z) \oplus i_2 \in LB(M\|m) \text{ for some} \\ &\qquad i_1\|i_2 \in I_j\}| \\ &\leq \sum_j |I_j| \times |LB(M\|m)| \leq 2|I| \leq 2(q_2 + q_3)\end{aligned}$$

3. $|B_4| \leq 1$ and $|B_5| \leq |I| \leq (q_2 + q_3)$
4. We partition I as before. Now,

$$\begin{aligned}|B_6| &= |\{z \in \{0,1\}^s : FH(y\|z) \oplus i_1 = m, LH(y\|z) \oplus i_2 = FH(e(i_1\|i_2)) \oplus x_1 \text{ for some} \\ &\qquad i_1\|i_2 \in I\}| \\ &= \sum_j |\{z \in \{0,1\}^s : FH(y\|z) \oplus i_1 = m, LH(y\|z) \oplus i_2 = FH(e(i_1\|i_2)) \oplus x_1 \text{ for some} \\ &\qquad i_1\|i_2 \in I_j\}| \\ &\leq \sum |I_j| = |I| \leq (q_2 + q_3)\end{aligned}$$

5. We partition I as before. Now,

$$\begin{aligned}&|B_7| \\ &\quad = |\{z \in \{0,1\}^s : FH(y\|z) \oplus i_1 = m, LH(y\|z) \oplus i_2 = FH(e(i_1\|i_2) \oplus i'_1 \text{ for some} \\ &\qquad i_1\|i_2, i'_1\|i'_2 \in I\}| \\ &\quad = \sum_j |\{z \in \{0,1\}^s : FH(y\|z) \oplus i_1 = m, LH(y\|z) \oplus i_2 = FH(e(i_1\|i_2) \oplus i'_1 \text{ for some} \\ &\qquad i_1\|i_2 \in I_j, i'_1\|i'_2 \in I\}| \leq \sum_j |I_j| \times |I| = |I|^2 \leq (q_2 + q_3)^2\end{aligned}$$

Hence all together we have

$$|B_1 \cup B_2 \cup \cdots \cup B_3| \leq (q_2 + q_3)^2 + 5(q_2 + q_3) + 3 \leq (q_2 + q_3 + 3)^2.$$

So, when $(q_2 + q_3 + 3)^2 < 2^s$ for any $y\|w \in \{0,1\}^{2n}$ we have,

$$\begin{aligned}\Pr[S^R(+,\cdot,\cdot) \textit{ query outputs } y\|w] &\leq \max\left(\frac{1}{2^{2n-s}} \frac{1}{2^s - (q_2+q_3+3)^2}, \frac{1}{2^{2n} - (q_2+q_3)}\right) \\ &\leq \frac{1}{2^{2n}} \frac{1}{1 - \frac{(q_2+q_3+3)^2}{2^s}}\end{aligned}$$

For $S^R(-,\cdot,\cdot)$ query giving output as $z_1\|z_2$, we know

- z_1 is uniformly distributed over $\{0,1\}^n \setminus C$
- z_2 is uniformly distributed over $\{0,1\}^n \setminus \{w \in \{0,1\}^n : z_1\|w \in I\}$

Initially size of C is 1 and during each query, the size of C can grow by at most $|I| + 1$ amount. Hence we have,

$$|C| \leq (q_2 + q_3)^2.$$

Also, by Lemma 4.2.2 we know,

$$|\{w \in \{0,1\}^n : z_1\|w \in I\}| \leq |I| \leq q_2 + q_3.$$

So, when $(q_2 + q_3)^2 \leq 2^n$ for any $z_1\|z_2 \in \{0,1\}^{2n}$ we have,

$$\Pr[S^R(-,\cdot,\cdot) \textit{ query outputs } z_1\|z_2] \leq \frac{1}{2^n - (q_2+q_3)^2} \frac{1}{2^n - (q_2+q_3)} \leq \frac{1}{2^{2n}} \frac{1}{1 - \frac{(q_2+q_3+1)^2}{2^n}}$$

Hence, all together we have

$$\Pr[\mathcal{OV}^{\mathcal{A}}_{R,S^R} = \mathcal{OV}] \leq \frac{1}{2^{(2n-s)q_1 + 2n(q_2+q_3)}} \times \frac{1}{(1 - \frac{(q_2+q_3+3)^2}{2^s})^{q_2}} \times \frac{1}{(1 - \frac{(q_2+q_3+1)^2}{2^n})^{q_3}}.$$

□

Theorem 4.5.2. *If* $2(q_2 + q_3 + 3)^2 < 2^{\min(s,n)}$*, the simulator* S^R *takes at most* $\mathcal{O}((q_2 + q_3)^2)$ *time to answer any query (except with exponentially negligible probability).*

4.5.2 Interpolation Probability of $\mathcal{OV}^{\mathcal{A}}_{JH^{\pi}_{P},\pi}$

The following theorem is analogous to Theorem 4.4.3, used in Section 4.4.

Theorem 4.5.3. *For any attacker* $\mathcal{A}$ *and any irreducible output view* $\mathcal{OV}$ *with respect to it, we have*

$$\Pr[\mathcal{OV}^{\mathcal{A}}_{JH^{\pi}_{P},\pi} = \mathcal{OV}] \geq \frac{1}{2^{(2n-s)q_1 + 2n(q_2+q_3)}} \times (1 - \frac{2\sigma^2}{2^{2n}}) \times (1 - \frac{2\sigma q_1(q_1+q_2+q_3)}{2^s}).$$

Proof. To prove the theorem we need an analog of Theorem 4.4.7 when chopping is done on the right-most (most significant) bits. We define the notion of CHOPR-matching in exact same way as of CHOPL$_s$-matching defined in Definition 4.4.6. Let $M_{\mathcal{V}}$ be the set of all such CHOPR-matching MD-irreducible tuples. Now we have the following theorem

Theorem 4.5.4. *For any irreducible view*

$$\mathcal{V} = ((M_1\|m_1, h_1), \dots, (M_{q_1}\|m_{q_1}, h_{q_1}), (x_1^1, x_1^2, y_1^1, y_1^2), \dots, (x^1_{q_2+q_3}, x^2_{q_2+q_3}, y^1_{q_2+q_3}, y^2_{q_2+q_3}))$$

we have,

$$|M_{\mathcal{V}}| \geq 2^{sq_1} \times (1 - \frac{2\sigma q_1(q_1+q_2+q_3)}{2^s}).$$

Proof. By Lemma 4.2.2, we know

$$\begin{aligned}&|\{z \in \{0,1\}^s : h_1\|z \oplus (m_1\|0^n) \in \{y_1^1\|y_1^2, \dots, y^1_{q_2+q_3}\|y^2_{q_2+q_3}\}\}|\\ &= |\{z \in \{0,1\}^s : h_1\|z \in \{y_1^1\|y_1^2 \oplus (m_1\|0^n), \dots, y^1_{q_2+q_3}\|y^2_{q_2+q_3} \oplus (m_1\|0^n)\}\}| \leq q_2 + q_3\end{aligned}$$

Also, we would like to have an upper bound for

$$|\{z \in \{0,1\}^s : h_1\|z \oplus 0^n\|m \in \{x_1^1\|x_1^2, \dots, x^1_{q_2+q_3}\|x^2_{q_2+q_3}, IV_1\|IV_2\} \text{ for some } m \in \Sigma\}|$$

Consider the case when $s \geq n$. We partition $\{x_1^1\|x_1^2, \dots, x^1_{q_2+q_3}\|x^2_{q_2+q_3}, IV_1\|IV_2\}$ as $S_1 \cup S_2 \cup \cdots$ such that for any $a, b \in \{x_1^1\|x_1^2, \dots, x^1_{q_2+q_3}\|x^2_{q_2+q_3}, IV_1\|IV_2\}$, a and b will go to the same partition (i.e. $a, b \in S_i$) iff $FH(a) = FH(b)$. Clearly,

$$\sum |S_i| = (q_2 + q_3 + 1).$$

Hence,

$$\begin{aligned}&|\{z \in \{0,1\}^s : h_1\|z \oplus 0^n\|m \in \{x_1^1\|x_1^2, \dots, x^1_{q_2+q_3}\|x^2_{q_2+q_3}, IV_1\|IV_2\} \text{ for some } m \in \Sigma\}|\\ &= \sum_i |\{z \in \{0,1\}^s : h_1\|z \oplus 0^n\|m \in S_i \text{ for some } m \in \Sigma\}| \leq \sum_i |S_i||\Sigma| \leq (q_2+q_3+1)\sigma\end{aligned}$$

In a similar way, we can also show

$$|\{z \in \{0,1\}^s : h_1\|z\oplus 0^n\|m \in \{x_1^1\|x_1^2, \dots, x^1_{q_2+q_3}\|x^2_{q_2+q_3}, IV_1\|IV_2\} \text{ for some } m \in \Sigma\}| \leq (q_2+q_3+1)\sigma$$

when $s < n$. Hence, there are at least $(2^s - (q_2 + q_3 + (q_2 + q_3 + 1)\sigma))$ many possible values for w_1. Once w_1 is selected there are at least $(2^s - (q_2 + q_3 + (q_2 + q_3 + 1)\sigma + 1))$ choices for w_2 and so on. Hence,

$$\begin{aligned}|M_{\mathcal{V}}| = &\text{ Number of valid } w \text{ tuples}\\ &\geq (2^s - (q_2 + q_3 + (q_2+q_3+1)\sigma)) \dots (2^s - (q_2+q_3+(q_2+q_3+1)\sigma + q_1 - 1))\\ &\geq 2^{sq_1} \times (1 - \frac{2\sigma q_1(q_1+q_2+q_3)}{2^s})\end{aligned}$$

□

We are now ready to prove Theorem 4.5.3. Let $\mathcal{V}$ be the irreducible view determined by $\mathcal{A}$ and irreducible output view $\mathcal{OV}$. Consider an Attacker $\mathcal{A}'$, which makes queries at the same input points as of $\mathcal{A}$, but has access to $MD^{f^{\mathcal{O}_2}}$ instead of JH_P^{π}.

$$\Pr[\mathcal{OV}^{\mathcal{A}}_{JH_P^{\pi},\pi} = \mathcal{OV}] = \sum_{v \in M_{\mathcal{V}}} \Pr[\mathcal{V}^{\mathcal{A}'}_{MD^{f^{\pi}},\pi} = v]$$
$$\geq \sum_{v \in M_{\mathcal{V}}} \frac{1}{2^{2nq_1+2n(q_2+q_3)}} \times (1 - \frac{2\sigma^2}{2^{2n}})$$
$$\geq \frac{1}{2^{2nq_1+2n(q_2+q_3)}} \times (1 - \frac{2\sigma^2}{2^{2n}}) \times 2^{sq_1} \times (1 - \frac{2\sigma q_1(q_1+q_2+q_3)}{2^s})$$
$$= \frac{1}{2^{(2n-s)q_1+2n(q_2+q_3)}} \times (1 - \frac{2\sigma^2}{2^{2n}}) \times (1 - \frac{2\sigma q_1(q_1+q_2+q_3)}{2^s})$$

The first inequality follows from Theorem 4.4.5 and the second inequality follows from Theorem 4.5.4.

□

4.5.3 Indifferentiability Security Bound

Theorem 4.5.5. *The JH_P^{π} mode of operation (with $(2n-s)$-bit output) based on a random permutation π is $(\mathcal{O}((q_2+q_3)^2), q_1, q_2+q_3, \epsilon)$ indifferentiable from a random oracle R, with*

$$\epsilon \leq \frac{2\sigma^2}{2^{2n}} + \frac{q_2(q_2+q_3+3)^2}{2^s} + \frac{q_3(q_2+q_3+1)^2}{2^n} + \frac{2\sigma q_1(q_1+q_2+q_3)}{2^s},$$

where σ is the maximum number of message blocks queried, q_1 is the maximum number queries to JH_P^{π} or R, q_2+q_3 is the maximum number of queries to π, π^{-1} or $S'^R(+,\cdot,\cdot), S^R(-,\cdot,\cdot)$. Here we also assume, $2(q_2+q_3+3)^2 < 2^{\min(s,n)}$.

Under reasonable assumptions, for an attacker making at most q queries with total σ many compression function invocations we have

$$\mathsf{Adv}_{\mathcal{A}} = \mathcal{O}(\frac{\sigma^2}{2^{2n}} + \frac{q^3}{2^n} + \frac{q^2\sigma}{2^s}).$$

4.6 Distinguisher $\mathcal{A}$ for JH without length padding at last block

Recall that the compression function of JH is based on a fixed permutation π. On input of the n-bit message block m and $2n$-bit chaining value $h_1||h_2$ the compression function outputs $f(m, h_1, h_2) = \pi(h_1, h_2 + m) + m||0^n$. JH applies chopped Merkle-Damgård transformation and outputs first t ($t = 2n - s$) bits of the output of final compression function. Here s denotes the number of chopped bits.

In case of JH-n, we have $s = n$. Our distinguisher first queries $h = C^{\pi}(M)$ with a random n-bit message M. The distinguisher appends 0^n with h and queries $t_1\|t_2 = \pi(+, h\|0^n)$. Note that when the distinguisher is interacting with (π, C^{π}), the second π query made by $C^{\pi}(M||M_2)$ will be on the

```
Distinguisher A

1. $M \in_R \{0,1\}^n$.
2. $h = \mathcal{O}_1(M)$.
3. $t_1 \| t_2 = \mathcal{O}_2(+, h \| 0^n)$.
4. $z_1 \| z_2 = \mathcal{O}_2(+, IV_1 \| IV_2 \oplus M)$.
5. $h_2 = \mathcal{O}_1(M \| z_2)$.
6. IF $t_1 \neq h_2 \oplus z_2$
       • return 1.
7. return 0.
```

(a) Distinguisher for JH-n without length padding

```
Distinguisher for $JH'$ without length padding at last block

• Choose distinct $n$-bit numbers $m_1, \ldots, m_k$
• For $i = 1, \ldots, k$
  $y_i^1 \| y_i^2 = \mathcal{O}_2(+, IV_1 \| IV_2 \oplus m_i)$
• If for $i = 1, \ldots, k$, $(y_1^i \oplus m_i)$'s are distinct return 1
• else
    – Find distinct $j_1, j_2$ such that $(y_{j_1}^1 \oplus m_{j_1}) = (y_{j_2}^1 \oplus m_{j_2})$
    – $m \in_R \{0,1\}^n$
    – $x_1 = \mathcal{O}_1(m_{j_1} \| (m \oplus y_{j_1}^2))$
    – $x_2 = \mathcal{O}_1(m_{j_2} \| (m \oplus y_{j_2}^2))$
    – if $x_1 \oplus \text{CHOPL}((m \oplus y_{j_1}^2) \| 0^n) \neq x_2 \oplus \text{CHOPL}((m \oplus y_{j_2}^2) \| 0^n)$
        * return 1
• return 0
```

(b) Distinguisher for JH' without length padding

Figure 4.6: The Distinguishers

input $(h||z)$ where z is the last n bit output of $\pi(+, IV_1, IV_2 \oplus M_1)$ xor-ed with M_2. So if we set M_2 to be the last n bit output of $\pi(+, IV_1, IV_2 \oplus M_1)$ then $z = 0^n$. Note that in case of JH with length padding, we could not choose M_2 this way as the length block is fixed. To get M_2, the distinguisher queries $z_1 \| z_2 = \mathcal{O}_2(+, IV_1 \| IV_2 \oplus M)$. Now $\mathcal{D}$ sets $M_2 = z_2$ and queries $h_2 = C^{\pi}(M \| z_2)$. Finally the distinguisher checks whether $h_2 = t_1 \oplus z_2$. Formal algorithm of the distinguisher is described in Figure 4.6(a).

Theorem 4.6.1. *If the simulator $\mathcal{S}$ makes at most k many $\mathcal{R}$ queries for answering a single query, then* $\mathsf{Adv}_{\mathcal{A}} \geq 1 - \frac{2k+1}{2^n}$

4.7 Distinguisher for JH'

In this section, we show one distinguisher with $\Omega(2^{n/2})$ many queries, which is successful against any simulator with non-negligible probability. Hence, when maximum query length ℓ is bounded by $2^{n/2}$, we get tight security bound.

The distinguisher has access to two oracles $\mathcal{O}_1, \mathcal{O}_2$ and is trying to differentiate between the two scenarios whether $(\mathcal{O}_1, \mathcal{O}_2)$ is (JH^{π}, π) or (R, S^R). Formal description of our distinguisher is given in Fig 4.6(b). The success probability of the distinguisher is established by following theorem.

Theorem 4.7.1. *With $k = \Omega(2^{n/2})$,* $\mathsf{Adv}_{\mathcal{A}}$ *is non-negligible for any simulator S.*

Proof. If there exists a simulator S against which $\mathcal{A}(k)$ has negligible advantage with $k = \Omega(2^{n/2})$, then the simulator must output a collision among $(y_1^1 \oplus m_1), \ldots, (y_k^1 \oplus m_k)$ with non-negligible probability. But the simulator also should find $y_{j_1}^2$ and $y_{j_2}^2$ such that the relation

$$R(m_{j_1} \| (m \oplus y_{j_1}^2)) \oplus \textsc{Chop}((m \oplus y_{j_1}^2) \| 0^n) = R(m_{j_2} \| (m \oplus y_{j_2}^2)) \oplus \textsc{Chop}((m \oplus y_{j_2}^2) \| 0^n)$$

holds with non negligible probability for $m \in_R \{0,1\}^n$. But R being a Random Oracle clearly that is not possible. $\square$

Note, if we use CHOPR instead of CHOPL then the same attack actually applies for the original JH mode of construction without length padding at last block as well.

Part II

Blackbox Separation of Padding Based Schemes

Chapter 5

Blackbox Separation of Padding Based Public Key Schemes: Definitions and Preliminaries

5.1 Introduction and Definitions

In public key cryptography, the keys of the sender and the receiver are different. Based on the application, either the sender or the receiver generates a pair of keys; a public key, which is known to everyone including the adversary and a secret key, which is only known to the person who generated the keys. Two of the most important applications of public key cryptography are encryption and signatures.

5.1.1 Public Key Encryption Scheme

A public key encryption scheme $\mathcal{E} = (\texttt{Gen}, \texttt{Enc}, \texttt{Dec})$ consists of three probabilistic polynomial time algorithms. The key generation algorithm Gen takes a security parameter 1^k as input and outputs an encryption key ek and a decryption key dk. We write $(pk, sk) \leftarrow \texttt{Gen}(1^k)$ to denote the execution of Gen. The (randomized) encryption algorithm Enc takes ek and the message M as input and outputs a ciphertext C. We denote the execution of the encryption algorithm by $C \leftarrow \texttt{Enc}(pk, M)$. The decryption algorithm (possibly randomized), on input dk and a cipher text C, outputs the message M (or a special symbol $\perp$ if C is an invalid ciphertext). We write $M \leftarrow \texttt{Dec}(sk, M)$ to denote an execution of the decryption algorithm.

Security against Chosen Plaintext Attack

A public key encryption scheme $\mathcal{E}$ is said to be *secure against chosen plaintext attack* (IND-CPA) if for any PPTM adversary $\mathcal{A}$, advantage of $\mathcal{A}$ in the following game is negligible in terms of the security parameter.

- $\texttt{Gen}(1^k)$ is executed to get (pk, sk). $\mathcal{A}$ gets 1^k and pk.
- $\mathcal{A}$ publishes two equal length messages M_0 and M_1. A bit $b \leftarrow \{0,1\}$ is chosen uniformly at random. $\mathcal{A}$ gets the challenge ciphertext $C = \texttt{Enc}(pk, M_b)$.
- $\mathcal{A}$ publishes a bit b' as output.

We say $\mathcal{A}$ to be successful if $b = b'$ and denote the probability of this event as $Pr^{CPA}_{\mathcal{A},\mathcal{E}}[Success]$. We define the advantage of $\mathcal{A}$ as $Adv^{CPA}_{\mathcal{A},\mathcal{E}} = |Pr^{CPA}_{\mathcal{A},\mathcal{E}}[Success] - \frac{1}{2}|$ where the probability is taken over the joint distribution of $\mathcal{A}$ and $\mathcal{E}$.

Security against Chosen Ciphertext Attack

A public key encryption scheme $\mathcal{E}$ is said to be *secure against chosen ciphertext attack* if for any PPTM adversary $\mathcal{A}$, advantage of $\mathcal{A}$ in the following game is negligible in terms of the security parameter.

- $\texttt{Gen}(1^k)$ is executed to get (pk, sk). $\mathcal{A}$ gets 1^k and pk.
- $\mathcal{A}$ makes at most polynomially many queries to the decryption oracle $\texttt{Dec}(sk, .)$.
- $\mathcal{A}$ publishes two equal length messages M_0 and M_1. A bit $b \leftarrow \{0,1\}$ is chosen uniformly at random. $\mathcal{A}$ gets the challenge ciphertext $C = \texttt{Enc}(pk, M_b)$.
- The adversary continues to make at most polynomially many queries to the decryption oracle $\texttt{Dec}(sk, .)$ except on the challenge ciphertext C.
- $\mathcal{A}$ publishes a bit b' as output.

$\mathcal{A}$ is said to be successful if $b = b'$ and we denote the probability of this event as $Pr^{CCA}_{\mathcal{A},\mathcal{E}}[Success]$. We define the advantage of $\mathcal{A}$ as $Adv^{CCA}_{\mathcal{A},\mathcal{E}} = |Pr^{CCA}_{\mathcal{A},\mathcal{E}}[Success] - \frac{1}{2}|$ where the probability is taken over the joint distribution of $\mathcal{A}$ and $\mathcal{E}$.

In this thesis we also consider signature schemes, another important public key cryptographic primitive.

5.2 Signature Schemes

A signature scheme $(\texttt{Gen}, \texttt{Sign}, \texttt{Verify})$ is defined as follows:

- The *key generation algorithm* $\texttt{Gen}$ is a probabilistic algorithm which given 1^k, outputs a pair of matching verification and signing keys, (vk, sk).

- The *signing algorithm* Sign takes the message M to be signed, the public key vk and the signing key sk, and returns a signature $\sigma = \mathtt{Sign}_{sk}(M)$. The signing algorithm may be probabilistic.

- The *verification algorithm* Verify takes a message M, a candidate signature σ' and vk. It returns a bit $\mathtt{Verify}_{vk}(M, \sigma')$, equal to one if the signature is accepted, and zero otherwise.

We require that if $\sigma \leftarrow \mathtt{Sign}_{sk}(M)$, then $\mathtt{Verify}_{vk}(M, \sigma) = 1$. We say a message signature pair (M, σ) to be valid if the signature $\sigma$is accepted for the message M.

5.2.1 Security of a Signature Scheme

In a secure Signature scheme, the security is ensured against an adversary whose objective is to *forge* a signature. The weakest security notion of a signature scheme is called unforgeability against zero message attack where adversary has to produce a valid message signature pair without even looking at the signature of any message. In stronger notions, adversary is allowed to ask (possibly adaptively) for signature of some messages of its choice and then produce a valid signature of a new message (whose signature was not queried).

Unforgeability against zero message attack

A signature scheme is said to be *unforgeable against zero message attack* if for any PPTM adversary $\mathcal{A}$, advantage of $\mathcal{A}$ in the following game is negligible in terms of the security parameter.

- $\mathtt{Gen}(1^k)$ is executed to get (vk, sk). $\mathcal{A}$ gets 1^k and vk.
- $\mathcal{A}$ publishes a message signature pair (M, σ).

We say $\mathcal{A}$ to be successful if $\mathtt{Verify}_{vk}(M, \sigma) = 1$.

Unforgeability against chosen message attack

A signature scheme is said to be *unforgeable against chosen message attack* (EUF-CMA) if for any PPTM adversary $\mathcal{A}$, advantage of $\mathcal{A}$ in the following game is negligible in terms of the security parameter.

- $\mathtt{Gen}(1^k)$ is executed to get (vk, sk). $\mathcal{A}$ gets 1^k and vk.
- $\mathcal{A}$ makes at most polynomially many queries to the signing oracle $\mathtt{Sign}(sk, .)$.
- $\mathcal{A}$ publishes a message signature pair (M, σ).

We say $\mathcal{A}$ to be successful if $\mathtt{Verify}_{vk}(M, \sigma) = 1$ and M was never queried to the signing oracle.

5.2.2 Padding Based Schemes

In a padding based public key scheme, a public, invertible padding is applied to the message (and the random string) before the operation (like encryption, signature etc). Examples of popular padding based encryption scheme includes OAEP [12], OAEP++ [105], SAEP [26], SAEP+ [26]. Similarly the Full Domain Hash and Probabilistic Signature Schemes can be considered as padding based signature schemes.

5.2.3 Trapdoor Permutations (TDPs)

Trapdoor Permutation is the most basic primitive of public key cryptography. In contrast to general notion of permutations, trapdoor permutations require a trapdoor to efficiently find the inverse.

Definition 5.2.1. A trapdoor permutation family is a triplet of PPTM $(Tdg, F\ , F^{-1})$. Tdg is probabilistic and on input 1^n outputs a key-pair $(pk, td) \leftarrow_R Tdg(1^n)$. $F(pk, .)$ implements a permutation π_{pk} over $\{0,1\}^n$ and $F^{-1}(td, .)$ implements the corresponding inverse π_{pk}^{-1}.

Security of Trapdoor Permutations

The most standard security property of TDP is *one-wayness* which says that it is hard to invert a random element without knowing the trapdoor. Formally, for any PPTM A

$$Pr[(pk, td) \leftarrow_R Tdg(1^n), x \leftarrow_R \{0,1\}^n : A(f_{pk}(x)) = x] \leq negl(n).$$

Many other security notions for Trapdoor Permutations are known. Like [43, 78], we consider a wide class of security properties using the notion of *δ-hard games*.

5.2.4 Hard Games

A cryptographic game consists of two PPTMs C (Challenger) and A (Prover) who can interact over a shared tape. After the interaction, C finally outputs a bit d. We say, A wins the game if $d = 1$ and denote it, following [43], by $\langle C, A \rangle = 1$.

Definition 5.2.2. [43] A game defined as above is called δ-hard game if for all PPT A (in the security parameter n) the probability of win , when both C and A have oracle access to t uniform random permutations $\pi_1, \pi_2, \cdots, \pi_t$ over $\{0,1\}^n$, is at most negligible more than δ. Formally C is a *δ-hard* game if for all PPTM A

$$\mathrm{Adv}_C(A, n) = Pr[\langle C^{\pi_1,\pi_2,\cdots,\pi_t}, A^{\pi_1,\pi_2,\cdots,\pi_t} \rangle = 1] \leq \delta + negl(n)$$

The hardness of the game C (denoted by $\delta(C)$) is the minimum δ such that C is δ-hard.

For cryptographic games like one-wayness, partial one-wayness, claw-freeness; $\delta = 0$. For the game of pseudo-randomness $\delta = 1/2$. The notion of δ-hard game was considered in [78] as a generalization of hard games considered in [43]. It was pointed out in [78] that the result of [43] can easily be extended to this notion.

5.2.5 Ideal Trapdoor Permutations

The notion of Ideal Trapdoor permutation was coined in [78]. To remain consistent with literature, we follow the same notion.

Let $TDP = (Tdg, F, F^{-1})$ be a trapdoor permutation. We say that TDP is secure for δ-hard game C if for all PPTM A, $\text{Adv}_C(A, n) - \delta(C)$ is negligible even when the random permutations in the definition of hard game is replaced by TDP. Formally, TDP is secure iff,

$$Pr[\langle C^{F(pk_1),F(pk_2),\cdots,F(pk_t)}, A^{F(pk_1),F(pk_2),\cdots,F(pk_t)}\rangle = 1] \leq \delta + negl(n),$$

where $(pk_i, td_i) \leftarrow_R Tdg(1^n)$ *for* $i = 1, \cdots t$.

Definition 5.2.3. [78] TDP is said to be an ideal trapdoor permutation if it is secure for any δ-hard game C.

We stress that, ideal trapdoor permutation does not exist. However as we are proving negative result in the thesis, proving separation from an ideal trapdoor permutation (hence to any hard game) makes our result stronger. This implies black-box separation from security notions like collision resistant hashing, pseudo-random functions, IND-CCA secure public key encryption schemes etc. In Fig. 5.1 we show some δ-hard game with the upper bound on advantage of any PPTM.

Game	Description of Game	Winning Condition	Advantage
One-wayness	$x \leftarrow_{\mathrm{R}} \{0,1\}^k; y \leftarrow F(pk_1, x); x' \leftarrow A^F(y)$	$x' = x$	$\leq \frac{q+1}{2^k}$
Partial Domain OW	$x \leftarrow_{\mathrm{R}} \{0,1\}^k; y \leftarrow F(pk_1, x); x' \leftarrow A^F(y)$	$x' = [x]_\ell$	$\leq \frac{q}{2^k} + \frac{2^k-\ell}{2^k-q}$
Claw-freeness	$(x_1, x_2) \leftarrow A^{F(pk_1,.),F(pk_2,.)}(1^k)$	$F(pk_1, x_1) = F(pk_2.x_2)$	$\frac{q^2}{2^k}$

Figure 5.1: Advantage of Ideal Trapdoor Permutations

5.2.6 Lossy Trapdoor Permutations

Lossy Trapdoor Functions were introduced by Peikert and Waters in [94]. In this chapter we consider a straightforward generalization to permutations. A family of (n, l) Lossy Trapdoor Permutations (LTDPs) is given by a tuple $(\mathcal{S}, \mathcal{F}, \mathcal{F}')$ of PPTMs. $\mathcal{S}$ is a sampling algorithm which on input 1 invokes $\mathcal{F}$ and on input 0 invokes $\mathcal{F}'$. $\mathcal{F}$ (called "Injective Mode") describes a usual trapdoor permutation; i.e. it outputs (π, π^{-1}) where π is a permutation over $\{0,1\}^n$ and π^{-1} is the corresponding inverse. $\mathcal{F}'$ (called

"Lossy Mode") outputs a function f' on $\{0,1\}^n$ with range size at most 2^l. For any distinguisher D, *LTDP-Advantage* is defined as

$$\text{Adv}^{ltdp}_{(\mathcal{F},\mathcal{F}'),D} = \left| Pr[D^{\pi}(.) = 1 : (\pi, \pi^{-1}) \leftarrow^{R} \mathcal{F}] - Pr[D^{f'}(.) = 1 : f' \leftarrow^{R} \mathcal{F}'] \right|.$$

We call $\mathcal{F}$ "lossy" if it is the first component of some lossy LTDP.

5.3 Existing work on the separation of padding based schemes

5.3.1 Blackbox Separation Results

A rich body of work [55, 56, 58, 63, 73, 107] on blackbox separation exists in the literature starting from the seminal work of Impagliazzo and Rudich [73]. Regarding the separation of random oracle from the standard model, the first result was due to Canetti, Goldreich and Halevi [30] who showed an artificial albeit valid signature scheme that cannot be securely instantiated by standard hash functions. Many such results [43, 50, 63, 78] were subsequently published.

To obtain our separation results we use the two oracle technique of Hsiao and Reyzin [71]. Specifically, we use the following generic version of proposition 1 of [71].

Proposition 5.3.1. *[71] To show that there is no black-box reduction from primitive $\mathcal{P}$ to secret-coin collision resistant hashing $\mathcal{Q}$, it suffices to construct two oracles T and G such that*

1. *There exists a PPTM M such that M^T implements $\mathcal{Q}$.*
2. *For all PPTM N, if N^T implements $\mathcal{P}$, there exists a PPTM A^G breaks the security of N^T.*
3. *For all PPTM B, B^G cannot break the security of M^T.*

5.3.2 Separation of Padding based schemes

The most relevant results to this thesis is the works of Dodis et. al. [43] and of Kiltz and Pietrzak [78]. In [43], Dodis, Oliviera and Pietrzak showed that the popular Full Domain Hash (FDH) signature scheme cannot be instantiated (using blackbox technique) in standard model by a ideal trapdoor permutation. Note that, FDH is a very special (deterministic) padding based signature scheme.

Theorem 5.3.2. *[43] There is no blackbox reduction of unforgeability against chosen message attack of Full Domain Hash signature scheme from any security property of ideal trapdoor permutation.*

In [93], Paillier showed impossibility of reduction of many RSA based signatures including PSS from different security assumptions of RSA. However, Paillier's impossibility result is based on an additional assumption (namely, instance non-malleability) of RSA.

Kiltz and Pietrzak [78] established that there is no blackbox reduction of any padding based CCA secure encryption scheme from ideal trapdoor permutations.

Theorem 5.3.3. *[78] There is no blackbox reduction of IND-CCA security of any padding based encryption scheme from any security property of ideal trapdoor permutation.*

Chapter 6

Impossibility of Instantiating PSS in the Standard Model

6.1 Introduction

Probabilistic Signature Scheme (PSS) is one of the most known and widely deployed provably secure randomized signature schemes. It was designed by Bellare and Rogaway [14] as a generic scheme based on a trapdoor permutation (like RSA). In [14], Bellare and Rogaway showed the scheme is secure in Random Oracle (RO) Model [12]. Coron improved the previous security bound in [34]. Recently in [37], PSS is proven secure even against fault attacks exploiting the Chinese Remainder Theorem (CRT) implementation of RSA . However, all the previous security proofs are valid only in RO model, where one assumes the existence of ideal, truly random hash functions. Unfortunately truly random functions do not exist and in practice, the "ideal" functions are instantiated with some efficient hash functions. Hence it is important that the proofs are valid while replacing random oracles by a standard hash functions. Otherwise such proofs merely provide heuristic evidence that breaking the scheme may be hard (or there is no generic attack against the scheme).

A number of papers [30, 43, 63, 78], starting from famous results of Canetti et. al. [30], showed that there are schemes secure in the Random Oracle model, which are uninstantiable under standard model. Naturally, these results raise concerns about the soundness of the schemes proven secure in random oracle model. Particularly for widely deployed scheme like PSS, it is especially important to have an secure instantiation by a standard, efficiently computable hash function so that we do not build our technology in vacuum. In this chapter, we ask essentially this particular question about PSS: *Whether it is possible, to securely instantiate PSS based on reasonable assumptions to the underlying trapdoor permutation.*

6.1.1 Our Results

Our main result is a general negative result to the above question. Roughly, we extend *all* the negative results by Dodis et. al. [43] for Full Domain Hash (FDH) to PSS. Specifically, we show the following

- There is no instantiation of PSS such that, *unforgeability under chosen message attack* can be reduced to any security property of a random permutation using *black-box* reduction techniques. As a random permutation satisfies almost all reasonable security notions, our result covers many of the standard security notions, like inverting trapdoor permutation on a random point (one-way), finding some bits of pre-image of a random point (partial domain one-wayness), finding correlated inputs etc. Our result is perfectly valid even if the hash functions used in PSS can query the trapdoor permutation and digests are arbitrarily related to the responses.

- We also rule out any black box reduction from recently proposed Lossy Trapdoor Permutations [94]. In Crypto 2010, Kiltz et. al. [77] has proven IND-CPA security of OAEP based on Lossy Trapdoor Permutation. Hence it is important to analyze whether positive result could be possible for PSS.

- We also show that even the weakest security criteria , namely unforgeability under *no message attack* cannot be black-box reduced to the one-wayness of the trapdoor permutation if the randomness space in PSS is "super-polynomial" in security parameter.

- All our results can easily be extended to the scenario when the adversary can invert some points of his choice (with some restrictions) for a fixed bounded number of times.

We would like to mention that our results does not completely rule out the possibility of instantiating PSS in standard model. A "whitebox" reduction, using the code of the adversary, may still exist. On the other hand, it may be possible to show a reduction from other cryptographic functions like homomorphic encryption. Still, we believe our result is important from theoretical point of view as it shows PSS requires special property of underlying trapdoor permutation as opposed to "Only randomness of hash is sufficient" notion of random oracle model.

Remark 6.1.1. All our results can be extended to any padding based signature scheme where the random string can be recovered.

6.1.2 Overview of our Technique

We use the technique of two oracles due to Hsio and Reyzin [71] for our separation results. We construct two oracles T and G such that T implements an ideal trapdoor permutation and G can be used to forge the PSS scheme. However, G does not help the attacker to break any security property of the ideal trapdoor permutation. Informally, this ensures that a black-box security proof cannot exist as any such proof should be valid against our T and G.

On a very high level our technique can be seen as an extension of the technique of Dodis et. al. [43] to rule out black box reduction of FDH. Separation from a random permutation is achieved in two steps. As the first step, we instantiate T by permutation chosen uniformly at random from the set of exponentially many permutations. Intuitively, G, the main forger oracle, should output a forgery after checking whether the adversary truly has access to a signer by sending polynomially many challenge messages. However the reduction could design the underlying hash function in such a way, so that the digests of the messages either collide with each other (hence reducing the number of points on which inversion is needed) or the digest is the result of one of the evaluation queries made to the trapdoor permutation (hence the reduction can get the signature from the corresponding query by evaluating the hash function). For this reason we define G to output the forgery only if the adversary can produce distinct signatures, which were not a query to the trapdoor permutation during the computation of digests, for all the challenge messages.

In the second step we show that a reduction algorithm (which does not have access to inversion oracle) cannot produce a valid signature meeting both the conditions with non-negligible probability. Hence to win any hard game, G is of no use to the adversary. However, we construct an efficient adversary with an access to a valid sign oracle (available in an unforgeability game) that can either find a forgery on its own or can construct signatures satisfying all the conditions of G. We stress that the efficient algorithm in [43], which precomputes all the hash values to check for the conditions, does not work efficiently when the signature scheme is randomized. Specifically, when the random strings are of super-logarithmic length, it is no longer possible for a polynomial time algorithm to compute all possible hash values for even a single message. It might very well happen that the computed digests meet the conditions but the digests on which signer generated the signature do not meet the condition. To solve this problem we use an elegant adaptive "evaluate on the fly" technique where we sample polynomially many random strings and check for the conditions. If the conditions are satisfied for the sampled digests, we repeatedly query the signer with fresh random coins for multiple signatures of same message. We show that, with probability exponentially close to 1, one gets either a set of valid signatures maintaining the conditions from the signer or could find a forgery during the sampling stage.

6.1.3 Differences from Crypto 05 paper of Dodis et. al.

Although our definition of oracles are quite similar to that in [43], difference comes in when finally implementing a forgery. The technique of [43] is not readily applicable for randomized signatures. Specifically in case of PSS the forger cannot force the signer to choose any particular random string. On the other hand, if the randomness space is super-polynomial the forger cannot pre-compute all the possible value of the hashes of any message. As a result the forger, as defined in [43], cannot output a forgery when G aborts. Our contribution is in constructing adaptive forger that can forge PSS with overwhelming probability even when the randomness space is super-polynomial. Moreover, our technique to rule out black-box reduction to one way trapdoor permutation is completely different. Looking ahead, we show that when the randomness space is of super-polynomial size, no Probabilistic Polynomial-Time Turing Machine (PPTM) can use a random signature (over the choice of random string during signing) of any fixed message to invert the one way trapdoor permutation.

6.1.4 Probabilistic Signature Scheme(PSS)

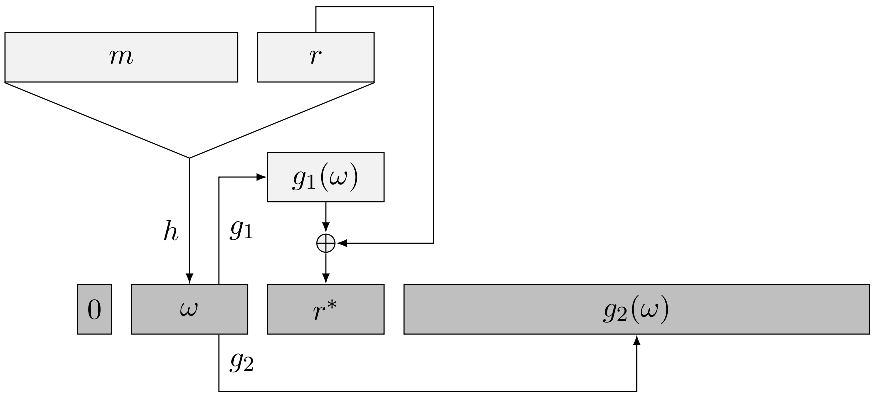

Figure 6.1: PSS_H^{TDP}: The components of the image $y = 0\|\omega\|r^*\|g_2(\omega)$ are darkened. The signature of m is $F^{-1}(td, y)$

Let $TDP = (Tdg, F, F^{-1})$ be a family of trapdoor permutations. PSS uses a triplet $H = (h, g_1, g_2)$ of hash functions such that, $h : \{0,1\}^* \to \{0,1\}^{k_1}$, $g_1 : \{0,1\}^{k_1} \to \{0,1\}^{k_0}$ and $g_2 : \{0,1\}^{k_1} \to \{0,1\}^{n-k_0-k_1-1}$, where n, k_0 and k_1 are parameters.
`Gen`(1^n)

1. Return $(pk, td) = Tdg(1^n)$

```
Sign_{td}(m)
    1. $r \leftarrow \{0,1\}^{k_0}$
    2. $\omega \leftarrow h(m\|r)$
    3. $r^* \leftarrow g_1(\omega) \oplus r$
    4. $y \leftarrow 0\|\omega\|r^*\|g_2(\omega)$
    5. Return $\sigma = F^{-1}(td, y)$.
```

```
Verify_{pk}(m, σ)
    1. Let $y = F(pk, \sigma)$
    2. Parse $y$ as $0\|\omega\|r^*\|\gamma$. If the parsing fails return 0.
    3. $r \leftarrow r^* \oplus g_1(\omega)$
    4. If $h(m\|r) = \omega$ and $g_2(\omega) = \gamma$ return 1.
    5. else return 0.
```

Any PSS signature scheme can be instantiated by specifying the triplet of hash functions $H = (h, g_1, g_2)$ and the trapdoor permutation TDP. Let PSS_H^{TDP} be the PSS signature scheme instantiated by H and TDP. For any $H = (h, g_1, g_2)$, the PSS transformation described above is defined as

$$PSS_H^{\pi_{pk}}(m\|r) = 0\|h(m\|r)\|(r \oplus g_1(h(m\|r))\|g_2(h(m\|r)).$$

$PSS_H^{\pi_{pk}}(m\|r)$ is in fact the darkened area in Figure 6.1, $y = 0\|\omega\|r^*\|g_2(\omega)$. *Note, h, g_1, g_2 can be oracle circuits with oracle access to π_{pk}.* For the rest of the chapter, PSS_H^{TDP} denotes the signature scheme, where as $PSS_H^{\pi_{pk}}(\cdot)$ is the PSS transformation during `Sign` procedure before applying the trapdoor permutation. From the context these two notations are easily distinguishable.

The following observation is very important to our technique.

Observation 6.1.2. *A collision after the PSS transformation implies collision in the random space. In other words,*

$$PSS_H^{\pi_{pk}}(M_1\|r_1) = PSS_H^{\pi_{pk}}(M_2\|r_2)$$

implies $r_1 = r_2$.

As both the digests are same, $\omega_1\|r_1^*\|\gamma_1 = \omega_2\|r_2^*\|\gamma_2$; we have $\omega_1 = \omega_2$ and $r_1^* = r_2^*$. This leads to $r_1 = r_2$. So for two distinct random strings r_1 and r_2, the digests of $PSS_H^{\pi_{pk}}$ and hence the signatures are always different (irrespective of whether the messages are same or not)! In case of other padding based signature schemes where random string is recoverable, then a similar condition can be found.

6.2 No Blackbox Reduction from One way Trapdoor Permutations

One-wayness is the most common security property of a trapdoor permutation. All the previous security proofs of PSS in Random Oracle model are based on one wayness of underlying trapdoor permutation (specifically RSA). In this section we consider the possibility of reducing security of PSS from one-wayness of a trapdoor permutation, but in standard model. We show that when $k_0 = \omega(\log n)$, one cannot prove PSS secure via a blackbox reduction from one way trapdoor permutation even if the forger is never allowed to query the signer.

Recall that, $r_1 \neq r_2$ implies $PSS_H^{\pi_{pk}}(0\|r_1) \neq PSS_H^{\pi_{pk}}(0\|r_2)$. So the set $\{PSS_H^{\pi_{pk}}(0\|r) | r \in \{0,1\}^{k_0}\}$ is of super-polynomial size. Even if G returns one random signature (from a choice of superpolynomially many) of message 0, it is unlikely to be of any use of the adversary intended to invert TDP^T on a uniformly chosen element z.

Following [71], Proposition 1, to rule out blackbox reductions, it is enough to construct two oracles T and G such the following holds:

- There exists an oracle PPTM TDP such that TDP^T implements a trapdoor permutation.
- There exist an oracle PPTM A such that $A^{T,G}$ finds a forgery under chosen message attack for $PSS_H^{TDP^T}$.
- TDP^T is an one-way trapdoor permutation relative to the oracles T and G. That is, TDP^T is an one-way permutation even if the adversary is given oracle access to T and G.

Definition of T

For any $n \in \mathbb{N}$, Choose $2^n + 1$ permutations $\pi_0, \pi_1, \pi_2, \cdots, \pi_{2^n-1}$ and g uniformly at random from the set of all permutations over $\{0,1\}^n$. Now the oracle T is defined as follows:

- $T_1(td) \to g(td)$ (generate public key from the trapdoor)
- $T_2(pk, y) \to \pi_{pk}(y)$ (evaluate)
- $T_3(td, z) \to \pi_{g(td)}^{-1}(z)$ (inversion)

Implementing TDP^T

We use $T = (T_1, T_2, T_3)$ in the following way to construct (in the functional sense) the trapdoor permutation $TDP^T = (Tdg, F_T, F_T^{-1})$.

- $Tdg(1^n)$ chooses a uniform random $td \leftarrow \{0,1\}^n$ and computes the corresponding public key as $pk = T_1(td)$ and outputs (td, pk).
- $F_T(pk, y)$ returns $T_2(pk, y)$.
- $F_T^{-1}(td, z)$ returns $T_3(td, z)$.

It is easy to check that as TDP^T implements a trapdoor permutation, as $g(td) = pk$.

Description of G

The oracle G takes as input $k \in \mathbb{N}$ and $H \in \{0,1\}^*$. G selects an $r \in_R \{0,1\}^{k_0}$ and returns $F_T^{-1}(td, PSS_H^{\pi_{pk}}(0\|r))$. Here (td, pk) are the output of $Tdg(1^k)$.

As G always outputs a forgery for message 0, we get the following result.

Lemma 6.2.1. *There is a PPTM A such that A^G outputs a forgery for PSS signature scheme.*

G does not break security of TDP^T

Next we shall prove that TDP^T is one way, even relative to G. This is not at all obvious as G always provides forgery of the form $\pi_{pk}^{-1}(PSS_H^{\pi_{pk}}(.))$ for a H of our choice! But we note that $G(.)$ samples one z' from a set of superpolynomial size and outputs $\pi_{pk}^{-1}(z)$. Even if the adversary sets $PSS_H^{\pi_{pk}}(0, r)$ for one r to be the challenge z she received, probability that $\pi_{pk}(G(.)) = z$ is negligible. On the other hand if $\pi_{pk}(G(.)) \neq z$, then knowledge of inverse of some other point does not help the adversary to find $\pi_{pk}^{-1}(z)$ with significant probability for a pseudorandom π_{pk}. Following the above discussion we have Lemma 6.2.2,

Lemma 6.2.2. *A random permutation $\pi : \{0,1\}^n \to \{0,1\}^n$ is one way even if adversary is allowed to make one inverse query on any input except the challenge.*

Proof. Suppose, for contradiction, the lemma is not true. Then there exist a PPTM $B = (B_1, B_2)$ such that

$$Pr[B_1^{\pi}(z) = z'; B_2^{\pi}(z, z', \pi^{-1}(z')) = x; z \neq z' : \pi(x) = z] \geq 1/n^c$$

for some constant $c > 0$. We construct a PPTM B' that , using B, can invert a random permutation $\pi' : T \to T$ where $|T| = 2^n - 1$. First we note that we can view T as the set $\{0,1\}^n \setminus 1^n$. B' keeps a list L with all the query responses. Without loss of generality, we assume B does not repeat a query and if B outputs some x, B must have queried it. Indeed, if B has not queried x, then the probability that $\pi(x) = z$ is negligible, which we can ignore.

Suppose B_1 and B_2 makes n^{c_1} and n^{c_2} queries respectively (c_1, c_2 are positive constant). B' works as following: on receiving the challenge z, check whether $z = 1^n$. If yes, abort; otherwise simulate $B_1(z)$. Clearly the probability that a randomly chosen z is equal to 1^n is $1/2^n$; hence negligible. When B_1 makes an oracle query x, check whether $x = 1^n$. If yes, select one element y_1 uniformly at random from $\{0,1\}^n \setminus L$. If $y_1 = z$; abort. Otherwise add $(1^n, y_1)$ to L. If $x \neq 1^n$, query $y = \pi'(x)$ and check whether $y = y_1$. If no add (x, y) to L. Otherwise add $(x, 1^n)$ to L. Clearly the probability that B' aborts at this stage is $1/(2^n - |L|)$. As B_1 makes only n^{c_1} number of queries, the above probability is negligible.

Suppose after all the queries, $B_1(z)$ outputs z'. Select one element x' uniformly at random from $\{0,1\}^n \setminus L$, add (x', z') to L and simulate $B_2(z, z', x')$. If B_2 makes an oracle query x ($x \neq 1^n$) s.t $\pi'(x) = z'$ compute $y' = \pi'(x')$. If $x' = 1^n$; $y' = 1^n$. Add (x, y') to L and proceed with y' as the answer. Otherwise if B_2 queries 1^n, select one element y_1 uniformly at random from $\{0,1\}^n \setminus L$. If $y_1 = z$; abort. Otherwise add $(1^n, y_1)$ to L. For all other query x compute $y = \pi(x)$. If $y = y_1$ reset $y = 1^n$. Add (x, y) to L.

Suppose B_2 returns x_1. Clearly $x_1 \neq 1^n$ as B' aborted whenever 1^n was queried and the result of the sampling was z. If $\pi'(x_1) = z'$ return x' from the list. else return x_1.

It is easy to check that, while answering the oracle queries B' simulates the random permutation. Moreover, if B_2 returns a correct answer so does B, except the case when B' aborts (when challenge is 1^n or B has queried 1^n and the result of the sampling was z). Probability that randomly chosen challenge is equal to 1^n is $1/2^n$. On the other hand, Probability that while sampling for the image of 1^n, z was picked is at most $\frac{1}{2^n - |L|} \leq \frac{1}{2^n - n^{c_1}}$ for some fixed constant c. So Probability that B' aborts is negligible. So Probability that B' inverts a randomly chosen z is at least $\frac{1}{n^c - negl(n)}$. This is a contradiction to the fact that a π' is hard to invert. Hence the lemma follows. □

Now, we can claim that TDP^T is one way even relative to G.

Lemma 6.2.3.

$$Pr[A^{T,G}(pk, z) = x : F_T(pk, x) = z] \leq negl(n),$$

where $x \leftarrow_R \{0,1\}^n$ *and* $(pk, td) \leftarrow Tdg(1^n)$.

Proof. As $F_T(pk, .)$ is a permutation chosen uniformly at random from a set of exponential size, $F_T(pk, .)$ is computationally indistinguishable from a random permutation. So if G was not there, TDP^T was clearly one way. Next we shall prove that TDP^T is one-way even when adversary has access to G. By the property of $PSS_H^{\pi_{pk}}$, there can be at most one $r \in \{0,1\}^{k_0}$ such that $PSS_H^{\pi_{pk}}(0||r) = z$. So Probability that G selects that corresponding r is $1/2^{k_0}$ which is negligible for $k_0 = \omega(\log n)$. On the other hand , if G does not select that particular r, by Lemma 6.2.2, a random permutation and hence $F_T(pk, .)$ (being computationally indistinguishable from a random permutation) is hard to invert . Hence

$$\begin{aligned}
&Pr[A^{T,G}(pk, z) = x : F_T(pk, x) = z] \\
&= Pr[A^{T,G}(pk, z) = x : F_T(pk, x) = z | G(\cdot) = F_T(td, z)].Pr[G(\cdot) = F_T(td, z)] \\
&+ Pr[A^{T,G}(pk, z) = x : F_T(pk, x) = z | G(\cdot) \neq F_T(td, z)] \cdot Pr[G(\cdot) \neq F_T(td, z)] \\
&\leq 1/2^{k_0} + (2^{k_0} - 1)/2^{k_0}.(negl(n)) \\
&= negl(n).
\end{aligned}$$

□

Using Lemma 6.2.1 and Lemma 6.2.3, we get the main result of this section as follows

Theorem 6.2.4. *There is no blackbox reduction of Security under no message attack of Probabilistic Signature Scheme with superpolynomial randomness space from Oneway Trapdoor Permutations.*

6.3 No Blackbox Reduction from an Ideal Trapdoor Permutation

The following theorem states that there is no adversary that can break the security of the TDP^T using any adversary (in black-box way) breaking $PSS_H^{TDP^T}$ by chosen message attack when TDP^T is an ideal permutation.

Theorem 6.3.1. *There is no black-box reduction from a family of ideal trapdoor permutations to the existential unforgeability against chosen message attack of the PSS signature scheme.*

Like the previous section, we shall construct a oracle G such that there exists a PPTM B such that B^G can forge PSS although TDP^T is secure even relative to G. We define T and TDP^T as in section 6.2.

Definition of G

The oracle G works as follows. On input the description of the hash function triplet $H = (h, g_1, g_2)$, and the security parameter n, it selects $t = \max(|H|, n)$ messages $m_1, m_2, \cdots m_t$ uniformly at random from $\{0,1\}^* \setminus \{0\}$ and outputs them as a set of challenge messages. G expects valid and *distinct* signatures of all the messages. G also keeps a list (initially empty) of description of input hash functions, the challenge messages and the forgery it returns. If the description of the hash matches then G outputs the same challenge messages. If it gets valid signatures (as described below) then it outputs the previously returned forgery from the list.

Once it receives the messages and the signatures $(m_1, m_2, \cdots, m_t, \sigma_1, \sigma_2, \cdots, \sigma_t)$, G checks for the following conditions.

1. $\sigma_1, \cdots, \sigma_t$ are valid signatures for $m_1, \cdots, m_t$. Recover $r_1, \cdots, r_t$ such that,
$$PSS_H^{\pi_{pk}}(m_1 \| r_1) = \pi_{pk}(\sigma_1), \cdots, PSS_H^{\pi_{pk}}(m_t \| r_t) = \pi_{pk}(\sigma_t).$$
2. $\sigma_i \neq \sigma_j$ (or equivalently $PSS_H^{\pi_{pk}}(m_i \| r_i) \neq PSS_H^{\pi_{pk}}(m_j \| r_j)$) for all $1 \leq i < j \leq t$.
3. $\{PSS_H^{\pi_{pk}}(m_1 \| r_1), \cdots, PSS_H^{\pi_{pk}}(m_t \| r_t)\} \cap Y_{\pi_{pk}}^{PSS_H}(r_1, \cdots, r_t) = \emptyset$ where
$$\begin{aligned} Y_{\pi_{pk}}^{PSS_H}(r_1, \cdots, r_t) = &\{\pi_{pk}(x) | \exists i,\ 1 \leq i \leq t, \\ &PSS_H^{\pi_{pk}}(m_i \| r_i) \text{ makes the oracle query } x\}. \end{aligned}$$

If all the above conditions are satisfied then G chooses one r uniformly at random from $\{0,1\}^{k_0}$ and returns $\pi_{pk}^{-1}(PSS_H^{\pi_{pk}}(0 \| r))$. Here pk is the public key generated by $Tdg(1^k)$.

G breaks the security of $PSS_H^{TDP^T}$

Lemma 6.3.2. *There is a PPTM B^G that can mount existential forgery by chosen message attack on PSS with overwhelming probability.*

Proof. The goal of B^G is to either generate a forgery on its own or use the `sign` oracle to get signatures of $m_1, \cdots, m_t$ such that Condition 1, Condition 2 and 3 get satisfied. Then B^G can use output of G to produce forgery for the message 0. We describe two constructions of B^G depending on size of the randomness space or k_0.

Case I: $k_0 = \mathcal{O}(\log n)$ **:**

In this case B^G precomputes $PSS_H^{\pi_{pk}}(m_i \| r)$ for all $r \in \{0,1\}^{k_0}$ and $i = 1 \cdots t$ and checks whether the Condition 3 from Section 6.3 would get satisfied or not for any possible choice of r by the `Sign` oracle. If the conditions are not satisfied, B can find some m_i, m_j, r_i, r_j, x such that

$$PSS_H^{\pi_{pk}}(m_i \| r_i) = \pi_{pk}(x),$$

where $PSS_H^{\pi_{pk}}(m_j \| r_j)$ makes the oracle query $\pi_{pk}(x)$. In this case B can easily produce the forged signature x for the message m_i.
Otherwise to take care of Condition 2, B^G calls the `Sign` oracle to get valid signatures for message m_i's one by one for $i = 1$ to t. After receiving the i^{th} signature σ_i it always recovers the randomness r_i and checks whether

$$PSS_H^{\pi_{pk}}(m_i \| r_i) = PSS_H^{\pi_{pk}}(m_j \| r_i)$$

for some $i < j \leq t$. Because of Observation 6.1.2 it is sufficient to check with the fixed r_i for collision detection purposes. If the above condition gets true again B^G can readily output a forged signature for message m_j as σ_i. Otherwise, B^G ends up with $\sigma_1, \cdots, \sigma_t$ such that all the three conditions in Section 6.3 are satisfied. So B^G can easily use G to produce a forgery for the message 0. Hence B^G succeeds to forge PSS with probability 1.

Case II: $k_0 = \omega(\log n)$ **:**

In this case the randomness space is of superpolynomial size, hence B^G cannot precompute all the possible outputs of $PSS_H^{\pi_{pk}}(m \| \cdot)$ even for a single message m. However, we observe that the "no collision" requirement or Condition 2 can easily be taken care of by a technique similar to the previous one. To take care of Condition 3, we adopt a sampling procedure. B^G works in two phases. In *Phase-I*, B samples some random r's from $\{0,1\}^{k_0}$ uniformly and simulate the signing procedure by the real `Sign` oracle that would be queried in *Phase-II*. Then the probabilities that Condition 3 gets satisfied in Phase-I or in Phase-II are essentially the same. We set our parameters such a way, with high probability either Condition 3 does not hold in Phase I (hence direct forgery) or it holds in Phase-II (forgery via oracle G, provided Condition 2 holds).

Algorithm 1 B^G : Phase-I

```
1: $r_i^k \leftarrow_R \{0,1\}^{k_0} : 1 \leq i \leq t, 1 \leq k \leq t$
2: $V = \{PSS_H^{\pi_{pk}}(m_i \| r_i^k) : 1 \leq i \leq t, 1 \leq k \leq t\}$
3: $Y = \{\pi_{pk}(x) | \exists i, k,\ 1 \leq i \leq t,\ 1 \leq k \leq t$
                         s. t. $PSS_H^{\pi_{pk}}(m_i, r_i^k)$ makes oracle query $\pi_{pk}(x)\}$
4: if $V \cap Y \neq \emptyset$ then
5:     Output Direct Forgery
6: end if
```

Success Probability of B^G in Case II

In Line 21 of Algorithm 2, Condition 1 and Condition 2 are always satisfied. So B^G can abort only in two ways.

1. In Line 16 of Algorithm 2, Σ_i becomes empty for some i, $1 \leq i \leq t$.

2. In Line 21 of Algorithm 2, Condition 3 gets violated. $r_1, \cdots, r_t$ be the random strings recovered from $\sigma_1, \cdots, \sigma_t$. Violation of Condition 3 over here implies there exist some i, j, $1 \leq i, j \leq t$, $i \neq j$ such that
$$PSS_H^{\pi_{pk}}(m_i \| r_i) = \pi_{pk}(x),$$
where $PSS_H^{\pi_{pk}}(m_j \| r_j)$ makes the oracle query x.

Moreover, in both the cases no forgery was found in Algorithm 1.

Let us consider the case where for some i, Σ_i is empty. It implies for some i, for all $k = 1, \cdots, t$, $\sigma_i^k \in X_{i,k}$ and hence was removed from Σ_i . Fix some i . Let us call the set of r for which $PSS_H^{\pi_{pk}}(m_i, r) = \pi_{pk}(x)$ and x was queried while computing $PSS_H^{\pi_{pk}}(m_i, r)$ as BAD. Suppose

$$Pr_r[r \in \text{BAD}] = \theta.$$

Now the event $\Sigma_i = \emptyset$ and no forgery was found in Phase-I implies that the random strings $r^{(i)}$, sampled in Phase 1 were not from the BAD set and all of $r_i^1, r_i^2, \cdots, r_i^t$ were from BAD. As `Sign` and B samples independently, probability of $\Sigma_i = \emptyset$ is $\theta^t(1-\theta)^t \leq 2^{-t}$. Taking union bound over all i, the probability that for some i, Σ_i is empty is at most $t/2^t$.

For the second case, the chosen σ_is were not queried while computing them; rather one σ_i was queried while computing some other σ_j. Recall that maximum number of π_{pk} queries (made by $PSS_H^{\pi_{pk}}$) while computing one signature is $|H|$. As, for any j $\Sigma_j \leq t$, for each $j = 1, 2, \cdots, t;\ j \neq i$, maximum number of π_{pk} queries made while computing Σ_j is at most $t|H|$. So overall, for all $j \neq i$, total number of π_{pk} queries made by the $PSS_H^{\pi_{pk}}$ was $t^2|H|$. As, there are $2^{|r|}$ choices of random string, implying $2^{|r|}$ choices for each σ_i^k, and `Sign` runs each time with independent random coins, probability that at least one σ_i^k was from those $t^2|H|$ many π_{pk} queries is at most $\frac{t^4}{2^{k_0}}$.

Algorithm 2 B^G : Phase-II

```
1:  for $i = 1$ to $t$ do
2:      $\sigma_i^1 \leftarrow \texttt{Sign}(m_i), \cdots, \sigma_i^t \leftarrow \texttt{Sign}(m_i)$
3:      $\Sigma_i = \{\sigma_i^1, \cdots, \sigma_i^t\}$
4:      Recover $r_i^1, \cdots, r_i^t$ from $\sigma_i^1, \cdots \sigma_i^t$ using $\texttt{Verify}$.
5:      for $j = i + 1$ to $t$ do
6:          if $PSS_H^{\pi_{pk}}(m_i \| r_i^k) == PSS_H^{\pi_{pk}}(m_j \| r_i^k)$ for some $1 \leq k \leq t$ then
7:              Output Direct Forgery $(m_j, \sigma_i^k)$
8:          end if
9:      end for
10:     for $k = 1$ to $t$ do
11:         $X_{i,k} \leftarrow \{x | PSS_H^{\pi_{pk}}(m_i \| r_i^k)$ makes oracle query $\pi_{pk}(x)\}$
12:         if $\sigma_i^k \in X_{i,k}$ then
13:             $\Sigma_i \leftarrow \Sigma_i \setminus \{\sigma_i^k\}$
14:         end if
15:     end for
16:     if $\Sigma_i = \emptyset$ then
17:         Output $\perp$
18:     end if
19:     Pick any $\sigma_i \in \Sigma_i$
20: end for
21: if $\sigma_1, \cdots, \sigma_t$ satisfy Condition 1, Condition 2 and Condition 3 from Section 6.3 then
22:     Output forgery via $G$
23: else
24:     Output $\perp$
25: end if
```

Hence we get that

$$\begin{aligned}
&\Pr[B^G \to \perp] \\
&\leq \Pr[\exists i; \Sigma_i = \emptyset] + \Pr[\exists i, j; \sigma_i \in \{\ \pi_{pk} \text{ queries made while computing } \sigma_j\ \}] \\
&\leq \frac{t}{2^t} + \frac{t^4 |H|}{2^{|r|}}
\end{aligned}$$

Putting $t = max(|H|, n)$, $|r| = \omega(\log n)$ and $|H| \leq n^c$ for some constant c, $\Pr[B^G \to \perp]$ is $negl(n)$.

□

G **does not break the security of** TDP^T

Lemma 6.3.3. *For any oracle* `PPTM` B *and any* δ*-hard game* C *(with* $t = t(n)$ *implicitly defined by* C*),*

$$Pr[\langle C^{F(pk_1),F(pk_2),\cdots,F(pk_t)}, A^{F(pk_1),F(pk_2),\cdots,F(pk_t),G}\rangle = 1] \leq \delta + negl(n),$$

where $(pk_i, td_i) \leftarrow_R Tdg(1^n)$ *for* $i = 1, \cdots t$.

Proof. The proof of the above lemma is essentially same as in proof of Lemma 2 in [43], where one argues in the absence of oracle G the claim holds because of computational indistinguishability of π_{pk} from a random permutation. Moreover, Lemma 6.3.4 below states the accepting condition of oracle G can only be satisfied with a negligible probability. □

Lemma 6.3.4. *Let* π *be a random permutation on* $\{0,1\}^n$ *and* $c \geq 1$ *be a constant,* $m_1, \cdots, m_t$ *be* n*-bit values with* $t = \max(|H|, n)$*. For any oracle* `TM` A *which makes at most* n^c *oracle queries, we have (the probability is over randomness of* π*)*

$$\Pr[A^\pi \to (H, x_1, \cdots, x_t)] = \mathtt{negl}(n)$$

where, $|H| \leq n^c$ *and the output satisfies the following conditions for some* k_0*-bit* $r_1, \cdots, r_t$

1. $\pi^{-1}(PSS_H^\pi(m_1\|r_1)) = x_1, \cdots, \pi^{-1}(PSS_H^\pi(m_t\|r_t)) = x_t$.
2. $\pi^{-1}(PSS_H^\pi(m_i\|r_i)) \neq \pi^{-1}(PSS_H^\pi(m_j\|r_j))$ *for all* $1 \leq i < j \leq t$.
3. $\{PSS_H^\pi(m_1\|r_1), \cdots, PSS_H^\pi(m_t\|r_t)\} \cap Y_\pi^{PSS_H}(r_1, \cdots, r_t) = \emptyset$*, where*

$$\begin{aligned} Y_\pi^{PSS_H}(r_1, \cdots, r_t) =& \{\pi(x) | \exists i, 1 \leq i \leq t, \\ & PSS_H^\pi(m_i\|r_i) \text{ makes the oracle query } x\}. \end{aligned}$$

Lemma 6.3.4 can proved following the same technique of Lemma 3 of [43]. For completeness we give the following proof.

Proof. Consider any oracle `TM` A, where A^π comes up with an output $(PSS_H, x_1, \cdots, x_t)$ after making n^c oracle queries. Even if A outputs only one x_i, which it did not query to the oracle π, then the probability that the relation $\pi^{-1}(PSS_H^\pi(m_i\|r_i)) = x_i$ holds for some r_i is negligible. Let X_π^A, $|X_\pi^A| = n^c$ denote all the oracle queries made by A^π, i.e.

$$X_\pi^A = \{x | A^\pi \text{ makes the oracle query } x\}.$$

Consider any fixed oracle circuit h, $|h| \leq n^c$ and k_0 bit values $r_1, \cdots, r_t$ satisfying condition 2 and 3. Let $X_\pi^h(r_1, \cdots, r_t) = \{\pi^{-1}(y) | y \in Y_\pi^h(r_1, \cdots, r_t)\}$, i.e.

$$X_\pi^h(r_1, \cdots, r_t) = \{x | \exists i, 1 \leq i \leq t, h^\pi(m_i\|r_i) \text{ makes the oracle query } x\}$$

and let

$$H(r_1, \cdots, r_t) = \{\pi^{-1}(h^{\pi}(m_1 \| r_1)), \cdots, \pi^{-1}(h^{\pi}(m_t \| r_t))\}.$$

Condition 3 states that $\pi(H(r_1, \cdots, r_t)) \cap \pi(X_{\pi}^{h}(r_1, \cdots, r_t)) = \phi$, and as π is permutation this is equivalent to

$$H(r_1, \cdots, r_t) \cap X_{\pi}^{h}(r_1, \cdots, r_t) = \phi.$$

Given $X_{\pi}^{h}(r_1, \cdots, r_t)$ and conditioned on h^{π} satisfies condition 3 for the fixed $r_1, \cdots, r_t$, the set $H(r_1, \cdots, r_t)$ is a random subset of $\{0,1\}^n \backslash X_{\pi}^{h}(r_1, \cdots, r_t)$. If condition 2 is satisfied then $H(r_1, \cdots, r_t) = t$, moreover $|X_{\pi}^{h}(r_1, \cdots, r_t)| \leq t|h| \leq tn^c$. Now the probability that $H(r_1, \cdots, r_t) \subseteq X_{\pi}^{A}$ can be upper bounded as

$$\begin{aligned}
\Pr[H(r_1, \cdots, r_t) \subseteq X_{\pi}^{A}] &= \prod_{i=0}^{t-1} \frac{|X_{\pi}^{A}| - |X_{\pi}^{A} \cap X_{\pi}^{h}(r_1, \cdots, r_t)| - i}{2^n - i - |X_{\pi}^{h}(r_1, \cdots, r_t)|} \\
&\leq \Big(\frac{|X_{\pi}^{A}|}{2^n - t - |X_{\pi}^{h}(r_1, \cdots, r_t)|}\Big)^t \\
&\leq \Big(\frac{n^c}{2^n - 2tn^c}\Big)^t.
\end{aligned}$$

By taking the union bound over all oracle circuits h, $|h| \leq n^c$ and all possible $r_1, \cdots, r_t$ we can now upper bound the probability that there exists some oracle circuit PSS_H, and k_0-bit values $r_1, \cdots, r_t$ satisfying condition 2 and 3 such that A^{π} have queried the x_i values satisfying condition 1 as

$$\sum_{|H|=1}^{n^c} 2^{|H|} \Big(\frac{n^c 2^{k_0}}{2^n - 2tn^c}\Big)^t.$$

As $t = \max(|H|, n)$ and assuming $k_0 < n - c\log n$ for all c and sufficiently large n,[1] we can easily show the above term is $\mathtt{negl}(n)$. □

6.4 No Reduction from Lossy Trapdoor Permutations

In this section we show that there is no blackbox reduction of existential unforgeability of PSS against chosen message attack from Lossy Trapdoor Permutations. Specifically, let $LTDP = (S, F, F')$ be a family of Lossy Trapdoor Permutations. We define the output of PSS based on LTDP as $\sigma = \pi^{-1}(PSS_H(m \| r))$ where $(\pi, \pi^{-1}) \in F$. Note that, while instantiating PSS by a lossy TDP, we consider the trapdoor permutation to be the injective mode of the TDP.

Theorem 6.4.1. *There is no blackbox reduction of existential unforgeability against chosen message attack of Probabilistic Signature Scheme from Lossy Trapdoor Permutations.*

Proof To prove Theorem 7.5.2, we need new definitions of the oracles.

[1]If $k_0 = n - c\log n$ for some constant c PSS is trivially insecure as a random n bit string will be a valid signature of message 0 with probability $2^{n - c\log n - n} = \frac{1}{n^c}$

Definition of $\mathcal{T}$

$\mathcal{T}$ is defined as a pair (T, T'). Choose $2^n + 1$ permutations $\pi_0, \cdots, \pi_{2^n-1}$ and g uniformly at random from the set of all permutations over $\{0,1\}^n$. Moreover choose 2^n functions $e_0, \cdots, e_{2^n-1}$ uniformly at random from the set of all functions from $\{0,1\}^n$ to $\{0,1\}^l$.

Oracle T works as follows:

- $T_1(td) \to g(td)$ (generate public key from the trapdoor)
- $T_2(pk, y) \to \pi_{pk}(y)$ (evaluate)
- $T_3(td, z) \to \pi^{-1}_{g(td)}(z)$ (inversion)

On the other hand T' is defined as follows

- $T'(pk, x) = \pi_{pk}(1^{n-l}||e_{pk}(x))$

Now we define the $LTDP^{T,T'} = (S, (F, F^{-1}), F')$ as follows

- $S(b)$ If $b = 1$, choose a uniform random $td \leftarrow \{0,1\}^n$ compute $pk = T_1(td)$ and return (pk, td), otherwise choose a uniform random $pk \leftarrow \{0,1\}^n$ and return $(pk, \perp)$.
- $F(pk, y)$ returns $T_2(pk, y)$.
- $F^{-1}(td, z)$ returns $T_3(td, z)$.
- $F'(pk, y)$ returns $T'(pk, x)$.

Lemma 6.4.2. *$LTDP^{T,T'}$ implements a secure (n, l) Lossy Trapdoor Permutation when $l = \mathcal{O}(n^{\frac{1}{c}})$ for a positive constant c.*

Proof. Recall that, to show the security of $LTDP^{T,T'}$, we need to argue that for any efficient distinguisher D, $|Pr[D^F = 1] - Pr[D^{F'} = 1]|$ is negligible. Consider a random function $e' : \{0,1\}^n \to \{0,1\}^l$ and a random permutation $\pi : \{0,1\}^n \to \{0,1\}^n$. It is easy to check that $\pi(1^{n-l}||e'())$ has the same distribution of a random permutation until a collision in e'. e' being a random function, the collision probability is $q^2/2^l$, which is negligible for $q = \mathcal{O}(n^{c_1}))$ for some constant $c_1 > 0$.

Now using the fact that a function (permutation) chosen uniformly at random from the set of exponentially many functions (permutations) is indistinguishable from a random function (permutation), the lemma follows. □

6.4.0.1 Definition of G

Intuitively, G will work exactly the same way as in the previous case when the underlying permutation is in injective mode. When the permutation is lossy G can abort instead of returning a forgery. So effectively, when instantiated by the lossy mode G always aborts and in injective mode G aborts if the conditions are not satisfied.

In more detail, G works in the following way. On input the description of the hash functions h, g_1 and g_2, it selects t (to be fixed later) messages $m_1, m_2, \cdots m_t$ uniformly at random from $\{0,1\}^* \setminus 0$ and outputs them as a set of challenge messages. G expects valid and *distinct* signatures of all the messages. G also keeps a list (initially empty) of description of input hash functions, the challenge messages and the forgery it returns. If the description of the hash matches then G outputs the same challenge messages. If it gets valid signatures (as described below) then it outputs the same forgery from the list.

Once it receives the messages and the signatures $(m_1, m_2, \cdots, m_t, \sigma_1, \sigma_2, \cdots, \sigma_t)$, G first checks whether the signatures are valid and distinct.

- $F(pk, \sigma_i) = PSS_H^{\pi_{pk}}(m_i \| r)$ for some r. This signature verification is to make sure that that calling algorithm has access to signing oracle.
- $\sigma_i \neq \sigma_j$ for all $i \neq j$

If the above two conditions are satisfied then G finds the random strings used in the signatures. Let $r_1, r_2, \cdots, r_t$ be the random strings

- $\{F(pk, \sigma_1), F(pk, \sigma_2), \cdots, F(pk, \sigma_t)\} \cap Y_T = \emptyset$ where

$$Y_T = \{F(pk, x) | \exists i, 1 \leq i \leq t,\ PSS_H^{\pi_{pk}}(m_i \| r_i) \text{ queries } F(pk, x)\}.$$

Finally G checks whether F is the lossy mode[1], if yes it aborts; otherwise G chooses one r uniformly at random from $\{0,1\}^{k_0}$ and computes the PSS hash of $0\|r$ as $y = 0\|h(0\|r)\|g_1(h(0\|r)) \oplus r\|g_2(h(0\|r))$. Finally it returns the forgery as $(0, F^{-1}(td, y))$.

In order to use G to distinguish the lossy and the injective mode, any distinguisher has to construct a satisfying assignment of G in injective mode. By Lemma 6.3.4, it happens with negligible probability and we get the following result.

Lemma 6.4.3. *Suppose* $k = \mathcal{O}(n^{\frac{1}{c}})$ *for a positive constant c.* $LTDP^{T,T'}$ *implements a secure* (n, k) *Lossy Trapdoor Permutation even relative to G.*

Existence of a forger B^G for PSS using the injective mode of the LTDP is satisfied by Lemma 6.3.2. This completes the proof of Theorem 7.5.2. □

[1] As description of F can be hardwired in G, G can easily check the mode of F by finding the possible inverses

6.5 No Reduction from Hard Games with Inversion

Like [43], our result can also be extended to the hard games with inversions. Informally, in a hard game with bounded inversion $\mathcal{C}$, the adversary is allowed to make polynomial $q(n)$ many inversion queries except on some points defined in the game (for one way game adversary is not allowed to make inversion queries on the challenge she received). Following [43], if we modify G to ask for signatures of $|H| + q(n)$ messages and modify Lemma 6.3.4 accordingly, we get the following two theorems.

Theorem 6.5.1. *There is no blackbox reduction of security against existential forgery under chosen message attack for PSS from any hard game with polynomial number of inversion queries.*

Theorem 6.5.2. *There is no blackbox reduction of security against existential forgery against zero message attack for PSS from an oneway trapdoor permutation, even with polynomial number of inversion queries.*

Chapter 7

Generic Insecurity of OAEP in the Seed Incompressibility model

7.1 Introduction

Optimal Asymmetric Encryption Padding (OAEP) is one of the most popular and widely deployed padding based encryption schemes. Proposed by Bellare and Rogaway in [12], OAEP has been part of numerous standards like PKCS#1 v2.1, ANSI X9.44, IEEE P1363 etc. Several variants of OAEP, such as OAEP++ by Shoup[105], Simplified OAEP(SAEP) of Boneh [26], exist in the literature. All these schemes can be categorized as padding based encryption scheme where a public, invertible, injective padding is applied to the message and the random string, before encrypting the result by some trapdoor permutation like RSA. For example, OAEP uses two round Feistel network involving two separate hash functions as padding. Decryption is done by first inverting the trapdoor permutation followed by inversion of the padding.

Due to its importance, a lot of work has been done on OAEP and other padding based encryption schemes. OAEP has been proved secure against Chosen Ciphertext Attack (IND-CCA) in [52] under the assumption that the trapdoor permutation is partial-domain one-way. OAEP++ has been proved IND-CCA secure in [105] assuming the underlying permutation is one-way. Recently, Kiltz, O'Neil and Smith [77] has proven *IND-CPA* security of OAEP from the assumption that trapdoor permutation is lossy. However, all the existing security results, except [77], were proved in the Random Oracle Model, where one assumes the existence of a truly random hash function. On the other hand, Kiltz and Pietrzak [78] have proved that no padding based encryption scheme can be proven IND-CCA secure by blackbox reduction from any security property satisfied by random permutation in the standard model.

Seed Incompressible Functions

Seed Incompressible Functions were introduced by Halevi, Myers and Rackoff in [65]. Informally, a seed incompressible function remains secure even if the adversary can have some information (size bounded above by fraction of keysize or some security parameter) of the hash function as an advice. One can view the notion of Seed Incompressibility as a generalization of Exposure Resilience where adversary cannot break Pseudorandomness of the hash function even if some part of the hash input was saved and used as advice. [65] showed that, in contrast to negative result of [63], Fiat-Shamir Transformation can be proved secure in the *standard model* if the underlying hash function is modeled as seed incompressible. Unfortunately the result of [78] remains valid even if the two hash functions in OAEP are assumed to be seed incompressible.

Our Results

If we closely observe the blackbox security proofs of IND-CCA security of OAEP in the random oracle model, one important argument in the proof is that the adversary cannot compute any ciphertext without explicitly querying the oracles. On the other hand, the main trick in the impossibility proof of [78] is that the reduction has no knowledge about the hash function evaluations, done by the adversary in order to construct the ciphertexts. This observation raises the question whether there is a reduction of IND-CCA security of OAEP, if the reduction can see some queries of the adversary, or, formulating in different terms, do the impossibility results hold if the adversary cannot even compute a single ciphertext on its own? In this chapter we investigate this question. From a broader perspective, we ask: *Can OAEP or other padding based encryption schemes can be proven IND-CCA secure in some model, significantly weaker than random oracle as well as stronger than standard model*?

In this chapter, we prove blackbox separation of IND-CCA security of OAEP from Ideal Trapdoor Permutations in a restricted model, called Seed Incompressibility model. Roughly speaking, Seed Incompressibility model can be viewed as a bridge between the Random Oracle and the standard model. Security game in the seed incompressibility model is carried out in two steps. In the first stage, like in the standard model, the adversary gets the description of the circuit computing the hash functions (padding). However, the adversary can save at most ϕ bits of information in the first stage. In the second phase the security game starts and the adversary, using the stored information and oracle access to the hash functions, attempts to win the security game. The Random Oracle model can be viewed only as the second stage of a seed incompressibility model where adversary cannot save any information in the first stage. On the other hand, the standard model can be considered as seed incompressible with superpolynomial storage. Using this model we formalize the idea of restricting the number of "private" queries made by the adversary.

- Extending the results of Kiltz and Pietrzak [78], we prove that, under the assumption that collision resistant hash functions exist, if the length of the advice string, $\phi > k(1+\frac{1}{c}) + (\log k)^2$ for some positive constant c, there is no blackbox reduction of IND-CCA security of OAEP with security parameter k and superpolynomial message space from ideal trapdoor permutation in the seed incompressibility model. This result implies that the separation holds even if the adversary can construct only one ciphertext on its own.

- We extend this result to the case where adversary cannot construct even one ciphertext without making queries to the oracles. Specifically, if the length of the random string is $\mu(r)$ and the length of the advice $\phi > (1+\frac{1}{c})\mu(r)$ for some positive constant c, there is no blackbox reduction of IND-CCA security of OAEP with superpolynomial message space from ideal trapdoor permutation in the seed incompressibility model.

- Finally, we consider a reduction from Lossy Trapdoor Permutations. We show that, if collision resistant hash functions exist and if the length of the random string is $\mu(r)$ and the length of the advice $\phi > (1+\frac{1}{c})\mu(r)$ for some positive constant c, there is no blackbox reduction of IND-CCA security of OAEP with superpolynomial message space from lossy trapdoor permutation in the seed incompressibility model.

We would like to mention that our result shows only blackbox separation of OAEP from a wide range of security properties. A whitebox security proof of OAEP may exist even in the standard model. However, we believe our result has strong theoretical interest as it hints a possible limitation in the structure of OAEP.

Overview of Our Technique

We use the technique of two oracles due to Hsio and Reyzin [71] for our separation results. We construct two oracles T and $B = (B_1, B_2)$ such that T implements an ideal trapdoor permutation and B can be used to break IND-CCA security of OAEP. Still, B does not help the reduction to break any security property of the ideal trapdoor permutation. Informally, this ensures that a black-box security proof cannot exist as any such proof should be valid against our T and B.

Our technique is almost similar to the technique of Kiltz and Pietrzak [78]. Our results on blackbox separation is achieved in two steps. At the first step, we instantiate T by permutation chosen uniformly at random from the set of exponentially many permutations. The adversary oracle B will output the secret key, only if the challenger can invert some ciphertexts created using the same public key. In case of IND-CCA game this requirement can easily be met with the help of the decryption oracle. On the other hand, if a reduction algorithm (which does not have access to any inversion oracle) does not know the exact queries made to the hash functions, it will not be able to decrypt the ciphertexts. However, in case of seed incompressibility model, if the length of the advice is small, the adversary algorithm needs to query the oracles in order to evaluate the padding scheme. The main challenge is to construct an adversary, which can create a challenge ciphertext in such a way that even after knowing some hash queries the reduction will not be able to decrypt the ciphertexts.

Informally, we show that in case of OAEP, if the message has superlogarithmic entropy, then the reduction even with the knowledge of queries made to the G oracle (first hash function of the two round Feistel network) cannot find the correct message from the ciphertext. We apply the compression argument of Gennaro and Trevisan [55] to show that if a polynomial size reduction could find the correct message with significant probability, the reduction could be used to compress a random permutation. We remark that this technique essentially solves the problem in the proof of [78], as pointed out by Dodis et. al.[42]. Although we got this result independently, the authors of [78] seem to have fixed the problem with essentially same technique[1].

Optimal Asymmetric Encryption Padding (OAEP)

Optimal Asymmetric Encryption Padding (OAEP), described in Figure 7.1, is parametrized by three parameters $k, \mu(r), k_1$ where $\mu(r), k_1$ is linear in terms of k. Let $F : \{0,1\}^k \to \{0,1\}^k$ be a family of trapdoor permutations and $G : \{0,1\}^{\mu(r)} \to \{0,1\}^{k-\mu(r)}$ and $H : \{0,1\}^{k-\mu(r)} \to \{0,1\}^{\mu(r)}$ be two cryptographic hash functions. The message space is $\{0,1\}^{k-\mu(r)-k_1}$. The OAEP Encryption Scheme $(\texttt{Gen}, \texttt{Enc}, \texttt{Dec})$ works in the following way,

- The Key generation algorithm $\texttt{Gen}(1^k)$ chooses $(\pi_{pk}, \pi_{pk}^{-1}) \in_{\text{R}} F$ and publishes the public key pk. Let sk be the corresponding secret key.

- The Encryption algorithm $\texttt{Enc}(pk, m)$ chooses a random string $r \in_{\text{R}} \{0,1\}^{\mu(r)}$ and computes

  - $s = m||0^{k_1} \oplus G(r)$
  - $t = H(s) \oplus r$

  Finally it outputs the ciphertext $C = \pi_{pk}(s\|t)$.

- The decryption algorithm $\texttt{Dec}(sk, C)$ recovers $s||t = \pi_{pk}^{-1}(C)$. Then it computes

  - $r = H(s) \oplus t$
  - $s' = G(r) \oplus s$ Finally it checks whether last k_1 bits of s are all 0. If yes, it parses $s' = m||0^{k_1}$ and outputs m. Otherwise it outputs $\perp$.

[1]Via Personal Communication

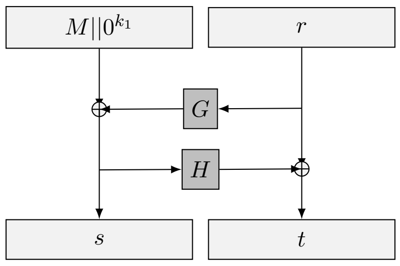

Figure 7.1: OAEP Transformation

Throughout the chapter we use $\texttt{OAEP}^F_{G,H}$ to represent OAEP encryption scheme instantiated by hash function G and H and the family of trapdoor permutations F. $\texttt{OAEP}_{G,H}(m, r) = s||t$ denotes the OAEP padding.

Seed Incompressible Functions

Seed Incompressible Functions was introduced by Halevi, Myers and Rackoff in [65]. Informally, a hash function family with keysize κ is called ϕ-Seed Incompressible function if for any adversary with $\phi(\kappa)$ bits of advice and an oracle access to the hash function cannot break any security property of the hash function with significantly more probability when given access to only the oracles. However, the idea is meaningless if the function is insecure even when the adversary is given only an oracle access. In this chapter, we consider the following definition of seed incompressible functions,

Definition 7.1.1. (**Seed Incompressible Functions[65]**) Let $\{h_\kappa\}_{\kappa\in\mathbb{N}}$ be a family of cryptographic hash functions. h_κ is called $\phi(\kappa)$ Seed Incompressible (SI) for the property Π, if

- For all PPTM $\mathcal{A}$,
$$Pr[\mathcal{A}^{h_\kappa} \text{ breaks security } \Pi] \leq \texttt{negl}(\kappa)$$
- For all pairs of PPTM $\mathcal{A}_1, \mathcal{A}_2$,
$$Pr[\mathcal{A}_1(\kappa) = \sigma, |\sigma| \leq \phi(\kappa); \mathcal{A}_2^{h_\kappa}(\sigma) \text{ breaks security}\Pi] \leq Pr[\mathcal{A}_2^{h_\kappa} \text{ breaks security}\Pi] + \texttt{negl}(\kappa)$$

This definition can also be extended to Seed Incompressible *Correlation Intractability* where the adversary (even with the advice) cannot satisfy some evasive relation.

Instantiation by seed incompressible functions

In [65], authors proved security of Fiat-Shamir Transformation, instantiated by seed incompressible correlation intractable function, in the standard model. Their reduction C works like the compressor of the seed incompressible function. They showed that if the adversary can break the soundness of the proof, then C can produce an advice σ, using which a PPTM $\mathcal{M}$ can break the correlation intractability of underlying seed incompressible function. On the other hand, the impossibility results of [78] in the standard model indeed cover possible instantiation by a seed incompressible function.

7.2 Seed Incompressibility model

In the Random Oracle model, the hash functions are modeled as random oracles. All parties including the adversary have only oracle access to the hash functions. In comparison, in the standard model, hash functions are assumed to be computable by efficient circuits. Adversary is given the description of the corresponding circuits as public parameters before the start of the security game.

In this chapter we consider the notion of seed incompressibility model. We argue that this model captures the concept of adversary's (partial) knowledge about the hash functions and bridges the gap between the Random Oracle and the standard model. In the seed incompressibility model, the adversary is a pair of algorithms $(\mathcal{A}_1, \mathcal{A}_2)$. $\mathcal{A}_1$ gets the description of the hash functions along with the public key. At the end of its execution $\mathcal{A}_1$ produces an advice σ to be used by $\mathcal{A}_2$. The size of the advice, $|\sigma|$, is upper bounded by the parameter ϕ. The security game is played with $\mathcal{A}_2$, which gets the public key as input and has only oracle access to the hash functions.

Definition 7.2.1. An oracle algorithm C is said to be $(\Pi \to \Pi',\ \phi, \delta, t)$ fully blackbox reduction for scheme S in seed incompressibility model, if for any seed incompressible Π adversary $(\mathcal{A}_1, \mathcal{A}_2)$, and for Π' candidate T, if

$$Adv^{\Pi}_{S^H[T]}(\mathcal{A}_1(\langle H \rangle) \to \sigma, |\sigma| \leq \phi : \mathcal{A}_2^H(\sigma)) \geq \epsilon$$

for a hash function H and constant $\epsilon > 0$, then

$$Adv^{\Pi'}_T(C^{T, \mathcal{A}^T}) \geq \delta(\epsilon, q)$$

where q is the total number of oracle queries made by C, t is the running time of C.

The following proposition holds for a particular hash function.

Proposition 7.2.2. *1. If there exists $(\Pi \to \Pi',\ 0, \delta, t)$ fully blackbox reduction for S in the seed incompressibility model, then there exists a $(\Pi \to \Pi', \delta, t)$-fully blackbox reduction for S in the random oracle model.*

2. If there exists $(\Pi \to \Pi',\ \delta, t)$ fully blackbox reduction for S in the standard model, then there exists a $(\Pi \to \Pi', |\langle H \rangle|, \delta, t)$-fully blackbox reduction for S in the seed incompressibility model.

We stress that, for a fully blackbox reduction in the seed incompressibility model, the reduction will not have access to any of the internal variable including the advice string used by the adversary. In both of the random oracle and the standard model, all the internal variables of the adversary are inaccessible to the fully blackbox reduction. As the definition of seed incompressibility model does not impose any restriction on the size of the advice string (restriction is imposed by the underlying seed incompressible function), one can view both the random oracle model (where size of the advice string is zero) and standard model (advice string can be of arbitrary length) as seed incompressibility model. Allowing direct access to the advice string will destroy this essence and the abstraction will be meaningless.

Notice that, the seed incompressibility model is essentially different to the Bounded-Retrieval Model [40, 46] where the adversary gets to know some amount of information about all the secrets (including the trapdoor). In the seed-incompressibility model, the adversary gets the description of only the hash function.

7.2.1 Blackbox Separation in Seed Incompressibility model

We use the two oracle technique of Hsiao and Reyzin [71] to disprove the existence of any fully blackbox reduction. Following [71], Proposition 1, to rule out blackbox reductions of OAEP to the ideal trapdoor permutation, it is enough to construct two oracles T and B such the following holds:

- There exists an oracle PPTM TDP such that TDP^T implements a trapdoor permutation.
- There exist an oracle PPTM A such that $A^{T,B}$ mounts a Chosen Ciphertext Attack on $\texttt{OAEP}^T_{G,H}$.
- TDP^T is an ideal trapdoor permutation relative to the oracles T and B. That is, TDP^T is an ideal trapdoor permutation even if the adversary is given oracle access to T and B.

Throughout the chapter we consider the following construction of T and TDP^T.

Definition of T

For any $n \in \mathbb{N}$, Choose $2^k + 1$ permutations $\pi_0, \pi_1, \pi_2, \cdots, \pi_{2^k-1}$ and g uniformly at random from the set of all permutations over $\{0,1\}^k$. Now the oracle T is defined as follows:

- $T_1(td) \to g(td)$ (generate public key from the trapdoor)
- $T_2(pk, y) \to \pi_{pk}(y)$ (evaluate)
- $T_3(td, z) \to \pi^{-1}_{g(td)}(z)$ (inversion)

Implementing TDP^T

We use $T = (T_1, T_2, T_3)$ in the following way to construct (in the functional sense) the trapdoor permutation $TDP^T = (Tdg, F_T, F_T^{-1})$.

- $Tdg(1^k)$ chooses a uniform random $td \leftarrow \{0,1\}^k$ and computes the corresponding public key as $pk = T_1(td)$ and outputs (td, pk).

- $F_T(pk, y)$ returns $T_2(pk, y)$.
- $F_T^{-1}(td, z)$ returns $T_3(td, z)$.

It is easy to check that as TDP^T implements a trapdoor permutation, as $g(td) = pk$.

7.3 No Blackbox Reduction for Long Advice: $\phi \geq k(1 + \frac{1}{c}) + (\log k)^2$

In this section, we prove that, if adversary is allowed to save $\phi \geq k(1 + \frac{1}{c}) + (\log k)^2$ bits of advice for any constant $c > 0$, there is no blackbox reduction of IND-CCA security of $\texttt{OAEP}^T_{G,H}$ from any security property of an ideal trapdoor permutation in the Seed Incompressibility model. Notice that size of the random string ($\mu(r)$) in OAEP has to be super-logarithmic in terms of the security parameter to have IND-CCA security. From Section 7.2.1, we need to construct a seed incompressible adversary $B = (B_1, B_2)$ that can break IND-CCA security of $\texttt{OAEP}^T_{G,H}$. Still TDP^T will remain secure even relative to B.

Description of B

Recall that, in seed incompressibility model the adversary works in two phases: first phase gets the description of the hash functions and saves an advice σ, the second phase using σ, breaks the security.

To meet this framework, B is divided in two parts; B_1 and B_2. Both B_1 and B_2 has same random function $h' : \{0,1\}^k \to \{0,1\}^{\frac{k}{c}}$. B_1 takes a description of the hash functions, picks a random message $m \leftarrow_{\text{R}} \{0,1\}^{(\log k)^2}$ and computes $\texttt{OAEP}_{G,H}$ on those messages with independently and uniformly selected random strings. [1] B_1 saves the message and the corresponding ciphertext. B_1 also saves $h'(pk)$ in σ. Note that, B_1 is stateless in the sense that it saves the same advice string if it is queried again with same input. B_2, on receiving the public key pk, first computes the $h'(pk)$ to check whether the same public key has been given. If yes, it computes the ciphertexts corresponding to the messages using pk and outputs them for decryption one by one. If the decrypted message matches with m, B_2 outputs the secret key corresponding to pk.

Formally B works in the following way

- **Phase I: Description of** B_1
  - On input $(k, pk, \langle G\rangle, \langle H\rangle)$, select $m \leftarrow_{\text{R}} \{0,1\}^{(\log k)^2}$ and $r \leftarrow_{\text{R}} \{0,1\}^{\mu(r)}$.
  - Compute $\alpha = h'(pk)$
  - Compute $c = \texttt{OAEP}^T_{G,H}(m \| 0^{k-\mu(r)-k_1-(\log k)^2}, r)$.

[1] We assume that message length is at least $(\log k)^2$. However, our proof works for any super logarithmic message length. We fix $(\log k)^2$ for notational convenience.

  - Save $\sigma = (\alpha \| m \| c)$.

- **Phase II: Description of B_2**
  - Ask for the public key pk.
  - If $h'(pk) \neq \alpha$ output $\perp$.
  - Parse σ as $(\alpha \| m \| c)$.
  - Output c for decryption.
  - If c is correctly decrypted, output $td = g^{-1}(pk)$. Otherwise output $\perp$

Breaking $\texttt{OAEP}^T_{G,H}$

It is straightforward to use B to break IND-CCA security of $\texttt{OAEP}^T_{G,H}$ in the seed incompressibility model. The adversary A, in the first phase , gets the public key pk and the description of the hash functions as input and initiates $B(k, \langle G \rangle, \langle H \rangle)$. When B asks for the public key, A will start the second phase and start the IND-CCA game. When A receives the public key, it will forward it to B to get the ciphertext c. Then it will query the decryption oracle to decrypt c and forward the decrypted message to B to get the secret key. Finally A will submit two random messages m_0, m_1 and can easily decrypt the challenge ciphertext by using the trapdoor and querying the G and H oracle. On the other hand, if B aborts, A also aborts.

Lemma 7.3.1. *There exists a PPTM A which with oracle access to B can break the IND-CCA security of $\texttt{OAEP}^T_{G,H}$.*

Note that, in case of OAEP, the message m and the random string r can be recovered from $s \| t$ by querying the oracle G and H. That is the reason we do not need to check whether $\texttt{OAEP}_{G,H}$ is already a IND-CCA scheme.

TDP^T is secure even relative to B

In this section, we prove that no efficient reduction can use B to break any security property of TDP^T.

Lemma 7.3.2. *If collision resistant hash functions exist, TDP^T implements an ideal trapdoor permutation, even relative to B.*

Proof. The fact that TDP^T is an ideal trapdoor permutation relative to T follows from Lemma 7 of [78]. By Lemma 10 of [78], to show that TDP^T is secure also relative to B, we need to show that B can be simulated by a polynomial time circuit. We show that if collision resistant hash function exists then there exists a simulator S, such that, for any PPTM C and for any δ hard game, output distribution of $C^{T,B}$ and C^{T,S^T} are computationally indistinguishable. S can be described in the following way.

S has a collision resistant hash function H. On input $(1^k, pk, \langle G\rangle, \langle H\rangle)$, S samples a message $m \leftarrow_{\text{R}} \{0,1\}^{(\log k)^2}$, and an $r \leftarrow_{\text{R}} \{0,1\}^{\mu(r)}$. If same input is given S repeats the same output. Then S computes the ciphertext by $c = \texttt{OAEP}^T_{G,H}(m\|0^{k-\mu(r)-k_1-(\log k)^2}, r)$ and the fingerprint $\alpha = h(pk)$. Finally, it saves $\alpha\|m\|c$.

In the second phase, the simulator on receiving the public key pk', first checks if $h(pk')$ is same as α.

- If $h(pk') \neq \alpha$, it outputs $\perp$. This perfectly simulates the B as h is collision resistant so C can produce two pk, pk' which collide with only a negligible probability. In case of B, h' is a random function and hence collision resistant against any polynomial size circuits.
- When C submits the message m', if $m \neq m'$, output $\perp$. In this case, C perfectly simulates B.
- If $m = m'$ output $\perp$. Note that this is the only case where S cannot simulate B perfectly.

We observe that, [78] also considers the case where C has made any F^{-1} queries. We stress that, in a hard game as defined in Chapter 5, the reduction does not have access to any F^{-1} oracle. So we can ignore the check for our case.

To bound the output difference of B and S, we need to show that for any reduction C, which queries B with input $(pk, \langle G\rangle, \langle H\rangle)$ and receives a ciphertext c for a randomly chosen message m and a randomly chosen string r, C can find m with only negligible probability. To show that, we prove the following lemma.

Lemma 7.3.3. *For any efficient circuit C making at most $poly(k)$ queries*

$$Prob[C(\texttt{OAEP}^T_{G,H}(m\|0^{\mu(m)-(\log k)^2}, r)) = m] \leq \texttt{negl}(k)$$

□

No reduction can recover m: Proof of Lemma 7.3.3

Fix $\mu(m) = (\log k)^2$. Let $\pi : \{0,1\}^k \to \{0,1\}^k$ be a fixed permutation. Let $\texttt{OAEP}_{G,H}$ makes at most l queries. Suppose, there exists an algorithm C which makes $q \leq k^{c_1}$ queries and a constant $c_0 > 0$

$$Prob_{m,r}[C(\pi(\texttt{OAEP}_{G,H}(m,r))) = m] \geq \frac{1}{k^{c_0}} \tag{7.1}$$

By averaging argument, there exists an $r \in \{0,1\}^{\mu(r)}$ such that

$$Prob_{m}[C(\pi(\texttt{OAEP}_{G,H}(m,r))) = m] \geq \frac{1}{k^{c_0}}$$

Let X_r be the set $\{(\texttt{OAEP}_{G,H}(m,r))|m \in \{0,1\}^{\mu(m)}\}$ and $Z = \{\pi(x)|x \in X_r\}$. Let $Y \subset Z$ be the set of elements for which C can correctly find out the corresponding m. We use the arguments of Gennaro and Trevisan [55] to argue that π can be described using significantly less than $\log(n-a)!$

bits. The only point we need to take care of is instead of saving the direct preimages of a large subset of Y, we need to save the corresponding m. Additionally we need to save responses of all the queries made by the $\texttt{OAEP}_{G,H}(m, r)$ in order to reconstruct the preimages. Formally the encoding works in the following way.

Encoding of π

We construct two sets S_z and S_m and a family of ordered list $\{R_i | i \in \{0,1\}^{\mu(m)}\}$. Initially all the sets are empty. We pick the lexicographically smallest element z of Y and put z in S_z. We simulate the computation of $C(z)$. Let $x_1, x_2, \cdots, x_q$ be the query made by $C(z)$ and $y_1, y_2, \cdots, y_q$ be the corresponding answers. If $y_i = z$ for some $i \in [q]$, define $\Gamma = Y \cap \{y_1, \cdots, y_{i-1}\}$. If $z \notin \{y_1, \cdots, y_q\}$, Γ is defined as $Y \cap \{y_1, \cdots, y_q\}$. Remove the set Γ from Y.

Let $C(z)$ returns a message m_z. Put m_z in S_m. Let $x'_1, x'_2, \cdots, x'_l$ be the sequential π queries made by $\texttt{OAEP}_{G,H}(m, r)$ and $y'_1, y'_2, \cdots, y'_l$ be the responses. If $y'_j = z$ for some j we remove $\{y'_1, y'_2, \cdots, y'_{i-1}\}$ from Y and add them to the list R_{m_z} maintaining the sequence. Otherwise we remove, $\{y'_1, y'_2, \cdots, y'_l\}$ from Y and add them to the list R_{m_z} maintaining the sequence. We pick the lexicographically smallest element remaining in the list and the whole procedure is repeated until Y is empty.

At each step one element is added to S_z and at most $k^{c_3} + 1$ elements are removed from X where $k^{c_3} = q + l$. So finally, both S_z and S_m will have at least $\frac{2^{\mu(m)}}{k^{c_0}(k^c+1)} \geq \frac{2^{\mu(m)}}{k^c}$ elements for a some constant c. As $\texttt{OAEP}_{G,H}$ makes at most l queries, each R_i will contain at most l elements each of k bits.

The sets S_z and S_m are kept as lists. for each $i \in \{0,1\}^{\mu(m)}$, R_i is saved as a table. Values of π on each element $\{0,1\}^k - \pi^{-1}(S)$ is written in a separate table.

Decoding of π

To decode π, first we decode the list L of preimages of the elements in S_z. For each $m \in S_m$ and simulate $\texttt{OAEP}_{G,H}(m, r)$ and return the answers of π queries reading sequentially from R_m. Moreover we add the corresponding queries and responses to the table (partially) describing π. If end of the list is reached and another query x is made then abort the simulation and put x in L. Otherwise $\texttt{OAEP}_{G,H}$ gets completed and outputs the x which is then added to the list. In both the cases we store $x = \texttt{OAEP}_{G,H}(m, r)$. Finally, L will have all the preimages of S_z.

After we construct L, value of π^{-1} on any element in $\{0,1\}^k - S_z$ can be found by reading from the table. The value of π^{-1} on elements from S_z are recovered in the lexicographic order. First pick the lexicographically smallest element z in S_z and simulate $C(z)$. If it makes a query $\pi(x)$, we know that either $x \in \{0,1\}^k - \pi^{-1}(S_z)$ (for which the value is written in the table) or $y = \pi(x) \in S_z$ and y is lexicographically smaller than z (hence the relation $y = \pi(x)$ has already been reconstructed and written in the table) or $z = \pi(x)$. In all of the cases, correct answers can given to C. Finally, when C outputs m, we find the corresponding $\pi^{-1}(z) = \texttt{OAEP}_{G,H}(m, r)$. So in all the cases we shall get the value of $\pi^{-1}(z)$.

The size of the encoding

Let $a = |S| \geq \frac{2^{\mu(m)}}{k^c} = 2^{\mu(m) - c\log k}$. The list S_z can be described using at most $\log \binom{2^k}{a}$ bits. The list S_m requires $\log \binom{2^{\mu(m)}}{a}$ bits. The ordered list R_is needs at most $a \times lk$ bits. Finally the rest of the permutation can be described using $\log(2^k - (l+1)a)!$ bits. So overall the π can be described using $\log \binom{2^k}{a} + \log \binom{2^{\mu(m)}}{a} + alk + \log(2^k - (l+1)a)!$ bits.

So the fraction of permutations for which Equation (7.1) holds true is

$$\frac{2^{alk}\binom{2^k}{a}\binom{2^{\mu(m)}}{a}(2^k - (l+1)a)!}{2^k!} = \frac{2^{alk}\binom{2^k}{a}\binom{2^{\mu(m)}}{a}(2^k - a)!}{2^k! \prod_{j=0}^{al-1}(2^k - a - j)}$$

$$= \frac{2^{alk}\binom{2^{\mu(m)}}{a}}{a! \prod_{j=0}^{al-1}(2^k - a - j)}$$

$$= \frac{\binom{2^{\mu(m)}}{a}}{a! \prod_{j=0}^{al-1}(1 - \frac{a+j}{2^k})}$$

As $al \leq 2^k$, for sufficiently large k, $\frac{1}{\prod_{j=0}^{al-1}(1 - \frac{a+j}{2^k})}$ can be bounded above by e. On the other hand $\frac{\binom{2^{\mu(m)}}{a}}{a!}$ can be upper bounded by

$$\left(\frac{4e^2 2^{\mu(m)}}{a^2}\right)^a \leq \left(\frac{4e^2 2^{\mu(m)}}{2^{2\mu(m) - 2c\log k}}\right)^a < 2^{-\frac{a\mu(m)}{2}}$$

In the last inequality we used $a = 2^{\mu(m) - c\log k} < 2^{\frac{\mu(m)}{2}}$.

Taking union bound over all circuits of size k^c, probability that Equation 7.1 holds against a randomly chosen permutation against any circuit is less than $2^{-\frac{a\mu(m)}{2} + ck^c \log k}$. As $\mu(m) = (\log k)^2$, $2^{\frac{-a\mu(m)}{2} + ck^c \log k}$ is negligible in terms of security parameter k.

Lemma 7.3.3 follows from the observation that $F_T(pk, .)$ is indistinguishable from a random permutation. $\square$

Putting it all together we get the main result of this section.

Theorem 7.3.4. *If the advice string can be of length $k(1 + \frac{1}{c}) + (\log k)^2$, there is no blackbox reduction of IND-CCA security of OAEP from ideal trapdoor permutation in the seed incompressibility model, assuming collision resistant hash functions exist.*

7.4 Impossibility for small randomness: $\phi(k) \geq \mu(r) + (\log k)^2 + \frac{k}{c}$

In this section we show that, if collision resistant hash function exists, $OAEP$ with small random string (say of size $\frac{k}{3}$) cannot be proven secure in the $\frac{k}{2}$-seed incompressibility model by black box reduction. Note that, if $\phi \leq \frac{k}{2}$, then an adversary cannot save even a single ciphertext as the advice. Hence the adversary needs to *query* the hash function oracles in order to create any ciphertext. As the queries made to the oracles are public, the reduction would be able to save the queries and their corresponding responses in a list. It may be possible (like the proofs in the random oracle model) that the reduction may be able to decrypt the challenge ciphertexts using these queries.

We show that, if there exists a collision resistant hash function $h : \{0,1\}^k \to \{0,1\}^{(\frac{k}{c})}$, then IND-CCA security of $\mathtt{OAEP}^F_{G,H}$ cannot be proven using blackbox reduction from ideal trapdoor permutations in the SI model. The idea is to compute $s\|t = \mathtt{OAEP}_{G,H}(m, r)$ for an m with entropy $(\log k)^2$ and a *fixed* random string r. We save m and t along with $\alpha = h'(pk)$ as advice where $h' : \{0,1\}^k \to \{0,1\}^{\frac{k}{c}}$ is a random function hardwired to B. α is saved to ensure that the reduction do not input a different public key in the second phase, which may make t invalid. Formally B works in the following way

- **Phase I**
  - On input $(k, pk, \langle G\rangle, \langle H\rangle)$, select $m \leftarrow_{\mathrm{R}} \{0,1\}^{(\log k)^2}$.
  - Set $r = 0^{\mu(r)}$
  - Compute $\alpha = h'(pk)$
  - Compute $s\|t = \mathtt{OAEP}_{G,H}(m\|0^{\mu(m)-(\log k)^2}, r)$.
  - Save $\sigma = (\alpha\|m\|t)$.
- **Phase II**
  - Ask for the public key pk.
  - If $\pi(pk) \neq \alpha$ output $\perp$.
  - Parse σ as $(\alpha\|m\|t)$.
  - Query G oracle with input $0^{\mu(r)}$.
  - Compute $s = G(0^{\mu(r)}) \oplus m\|0^{k-\mu(r)-(\log k)^2}$.
  - Compute the ciphertext $c = F_T(pk, s, t)$
  - Output c for decryption.
  - If c is correctly decrypted, output $td = g^{-1}(pk)$. Otherwise output $\perp$

Breaking $\texttt{OAEP}^T_{G,H}$ with small randomness

The adversary A to break the security of $\texttt{OAEP}^T_{G,H}$ works exactly as in Section 7.3. On receiving the public key and the description of $\texttt{OAEP}_{G,H}$, A queries B to get the ciphertext, decrypt it using the decryption oracle of IND-CCA game and return it to B to get the trapdoor td. Then A will be able to correctly decrypt the challenge ciphertext and win the game.

Lemma 7.4.1. *There exists a PPTM A which with oracle access to B can break the IND-CCA security of $\texttt{OAEP}^T_{G,H}$ with small randomness with probability* 1.

TDP^T is secure even relative to B

In this section, we prove that no efficient reduction can use B to break any security property of TDP^T.

Lemma 7.4.2. *If collision resistant hash functions exist, TDP^T implements an ideal trapdoor permutation even relative to B.*

Proof. The proof works exactly the same way as in the proof of Lemma 7.3.2. There are only two major difference to the previous case. First, the ciphertext is computed in the second phase and the reduction gets to see the G query and its output. We claim that m has superlogarithmic entropy even relative to r and $G(r)$. This might seem incorrect as m was chosen based on the public key and description of G and H. We argue that B uses a (hardwired) uniform random function to find m. For a uniform random function output distribution is independent to the input distribution. That would ensure the necessary binding to pk, G and H as well as the required independence.

The second difference is that r is fixed instead of being chosen uniformly at random. However, Lemma 7.3.3 is essentially proved for a fixed r and a message with $(\log k)^2$ entropy.

Lemma 7.4.3. *For any efficient circuit C making at most $poly(k)$ queries*

$$Prob_{m,C}[C(\texttt{OAEP}^T_{G,H}(m,r))) = m] \leq \texttt{negl}(k)$$

Hence the claim follows. ☐

We get the main result of this section.

Theorem 7.4.4. *If $\phi \geq \mu(r) + (\log k)^2 + \frac{k}{c}$, there is no blackbox reduction of IND-CCA security of OAEP from an ideal trapdoor permutation in the seed incompressibility model*

7.5 No Reduction from Lossy Trapdoor Permutations

In this section we show that there is no blackbox reduction of IND-CCA security of OAEP from Lossy Trapdoor Permutations even in seed incompressibility model. Specifically, Let $LTDP = (S, F, F')$ be a family of Lossy Trapdoor Permutations. We define the output of OAEP based on LTDP as $c =$

$\pi(\texttt{OAEP}_{G,H}(m,r))$ where $(\pi, \pi^{-1}) \in F$. Note that, while instantiating OAEP by a lossy TDP, we consider the trapdoor permutation to be the injective mode of the TDP. Otherwise, the decryption cannot be uniquely defined.

Theorem 7.5.1. *Assuming collision resistant hash functions exist, if $\phi \geq k(1+\frac{1}{c}) + (\log k)^2$, there is no blackbox reduction of IND-CCA security of OAEP from lossy trapdoor permutation in the seed incompressibility model.*

Theorem 7.5.2. *Assuming collision resistant hash functions exist, if $\phi \geq \mu(r) + (\log k)^2 + \frac{k}{c}$, there is no blackbox reduction of IND-CCA security of OAEP from a lossy trapdoor permutation in the seed incompressibility model.*

To prove Theorem 7.5.1 and Theorem 7.5.2, we need new definitions of the oracles.

Definition of T

The definition of the oracle is same as in the previous chapter. We recall the description here for completeness. $\mathcal{T}$ is defined as a pair (T, T'). Choose $2^k + 1$ permutations $\pi_0, \cdots, \pi_{2^k-1}$ and g uniformly at random from the set of all permutations over $\{0,1\}^k$. Moreover choose 2^k functions $e_0, \cdots, e_{2^k-1}$ uniformly at random from the set of all functions from $\{0,1\}^k$ to $\{0,1\}^l$.

Oracle T works as follows:

- $T_1(td) \to g(td)$ (generate public key from the trapdoor)
- $T_2(pk, y) \to \pi_{pk}(y)$ (evaluate)
- $T_3(td, z) \to \pi^{-1}_{g(td)}(z)$ (inversion)

On the other hand T' is defined as follows

- $T'(pk, x) = \pi_{pk}(1^{n-l}||e_{pk}(x))$

Now we define the $LTDP^{T,T'} = (S, (F, F^{-1}), F')$ as follows

- $S(b)$ If $b = 1$, choose a uniform random $td \leftarrow \{0,1\}^k$ compute $pk = T_1(td)$ and return (pk, td), otherwise choose a uniform random $pk \leftarrow \{0,1\}^k$ and return $(pk, \perp)$.
- $F(pk, y)$ returns $T_2(pk, y)$.
- $F^{-1}(td, z)$ returns $T_3(td, z)$.
- $F'(pk, y)$ returns $T'(pk, x)$.

The adversary B

we show that, if collision resistant hash function exists, $OAEP$ with small random string (say of size $\frac{k}{3}$) cannot be proven secure in the $\frac{k}{2}$-seed incompressibility model by black box reduction to lossy trapdoor permutation. Informally B will work exactly in the same way as in Section 7.4 except that in the last step, after getting the correct m, B_2 will check whether $\mathcal{T}$ is in lossy mode. If yes, B_2 aborts, otherwise B_2 publishes the corresponding secret key.

$LTDP^{\mathcal{T}}$ is secure relative to B

The simulator works exactly in the same way as in previous section. The output distribution of $C^{\mathcal{T},B}$ and $C^{\mathcal{T},S^{\mathcal{T}}}$, are indistinguishable, S simulates B perfectly when $\mathcal{T}$ is in lossy mode. When $\mathcal{T}$ is in injective mode, Lemma 7.3.3 implies that no polynomial size circuit can return correct m with non-negligible probability. This proves Theorem 7.5.1 and Theorem 7.5.2. $\square$

Chapter 8

Conclusion and Open Problems

In this thesis, we presented some results on blackbox reduction and separation of popular cryptographic constructions. We considered both symmetric and public key constructions.

In symmetric key, we considered indifferentiability of different modes of operation of cryptographic hash functions. We proposed a unified method to prove indifferentiability of a wide class of iterated hash functions, called GDE. Using our method we proved optimal indifferentiability bounds for Merkle-Damgård construction with counter (e.g. HAIFA) mode and for Tree Mode constructions with a similar sequential padding. This result shows tight indifferentiability bound (when the underlying compression functions are realized as random oracles) for many SHA3 candidates like BLAKE, LANE, SHAvite-3, MD6 etc. We also considered the security of a SHA 3 second round candidate, JH, in the indifferentiability framework. We showed that under the assumption that the underlying permutation is a random permutation, JH mode of operation with specific padding rule is indifferentiable from a Random Oracle. We also considered a modified design of JH, called JH$'$, by chopping different bits. We analyzed the indifferentiability of JH$'$ mode with optimal bounds. We also presented a distinguisher for JH mode without length padding (with any other prefix free padding). However, our attack does not pose any serious threat to JH hash function.

In the public key setting, we proved blackbox separation of popular padding based encryption and signature schemes. We showed that PSS or any padding based signature scheme (where the random string can be recovered from the signature) cannot be proven existentially unforgeable in the standard model using blackbox technique under a large class of cryptographic assumptions. We also proved that even if we restrict the adversary to a model where it has only limited information about the underlying hash functions, OAEP cannot be proven IND-CCA secure using a blackbox reduction.

8.1 Open Problems

This thesis leaves a number of open problems and interesting research directions. We list some of the possible directions below.

- The Envelope Merkle-Damgård construction of Bellare and Ristenpart can be seen to be in the class of Generalized Domain Extension. It would be highly interesting to see whether our method can be used to prove optimal indifferentiability bound of Envelope MD as well. In general, it is interesting to explore whether our technique can be used to prove improved bound for any prefix free Merkle-Damgård domain extension.

- In the definition of GDE we considered the underlying object to be a non-invertible random function. Extending our method to invertible objects (ideal cipher or random permutation) will also be an interesting direction.

- Our work on GDE presents sufficient condition of indifferentiability. Finding necessary condition will also be worthwhile to attack.

- To prove indifferentiability of JH we formalized an interpolation probability based technique. It is natural to explore whether a game based proof can be found to obtain similar bound.

- Improving our indifferentiability bound of JH remains a fascinating open problem due to its selection to the final round of SHA3 competition. However, such an analysis seems to need substantial insight.

- It seems that JH mode can also be used for authenticated encryption and online encryption. It may be interesting to pursue research in those directions.

- JH$'$ mode has a similar indifferentiability bound as sponge. It is interesting to compare these two constructions from other security aspects (like collision resistance, preimage resistance) as well.

- It would be interesting to explore that whether the security of PSS or any other padding based signature scheme can be proven in the generic group model where every party has oracle access to a group isomorphic to the underlying groups. Recently Dodis et. al. has made some progress towards this direction.

- It may be an interesting open problem to analyze whether one can prove IND-CCA security of OAEP in the seed incompressibility model where the size of the advice is significantly smaller than the random string. Instantiating oracle G by pseudorandom generators and H by MAC may be a good starting point.

- Exploring the power of seed incompressibility model will definitely remain an interesting research direction.

Bibliography

[1] Masayuki Abe, Eike Kiltz, and Tatsuaki Okamoto. **Chosen Ciphertext Security with Optimal Ciphertext Overhead**. In *ASIACRYPT*, pages 355–371, 2008.

[2] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. **On the Security of Joint Signature and Encryption**. In *EUROCRYPT*, pages 83–107, 2002.

[3] Elena Andreeva, Bart Mennink, and Bart Preneel. **Security Reductions of the Second Round SHA-3 Candidates**. Cryptology ePrint Archive, Report 2010/381, 2010. `http://eprint.iacr.org/`.

[4] Elena Andreeva, Bart Mennink, and Bart Preneel. **The parazoa family: Generalizing the sponge hash functions.** Cryptology ePrint Archive, Report 2011/028, 2011.

[5] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and Raphael C.-W. Phan. **SHA-3 proposal BLAKE**. Submission to NIST (Round 1/2), 2008.

[6] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and Raphael C.-W. Phan. **SHA-3 proposal BLAKE:Revision**. Submission to NIST (Round 3), 2010.

[7] Boaz Barak. **Non-Black-Box Techniques in Cryptography**. PhD thesis, EECS Department, University of California, Berkeley, 2004.

[8] R. Barke. **On the Security of Iterated MACs**. Diploma thesis, ETH Zurich, 2003.

[9] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. **Relations Among Notions of Security for Public-Key Encryption Schemes**. In *CRYPTO*, pages 26–45, 1998.

[10] Mihir Bellare and Thomas Ristenpart. **Multi-Property-Preserving Hash Domain Extension and the EMD Transform**. In *ASIACRYPT*, pages 299–314, 2006.

[11] Mihir Bellare and Thomas Ristenpart. **Multi-Property-Preserving Hash Domain Extension and the EMD Transform**. In *ASIACRYPT*, pages 299–314, 2006.

[12] Mihir Bellare and Phillip Rogaway. **Random Oracles are Practical: A Paradigm for Designing Efficient Protocols**. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[13] Mihir Bellare and Phillip Rogaway. **Optimal Asymmetric Encryption**. In *EUROCRYPT*, pages 92–111, 1994.

[14] Mihir Bellare and Phillip Rogaway. **The Exact Security of Digital Signatures - HOw to Sign with RSA and Rabin**. In *EUROCRYPT*, pages 399–416, 1996.

[15] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. **On the Indifferentiability of the Sponge Construction**. In *EUROCRYPT*, pages 181–197, 2008.

[16] Rishiraj Bhattacharyya. **Instantiating OAEP by seed incompressible functions**, 2011.

[17] Rishiraj Bhattacharyya and Avradip Mandal. **On the Impossibility of Instantiating PSS in the Standard Model**. In *Public Key Cryptography*, pages 351–368, 2011.

[18] Rishiraj Bhattacharyya, Avradip Mandal, and Mridul Nandi. **Indifferentiability Characterization of Hash Functions and Optimal Bounds of Popular Domain Extensions**. In *INDOCRYPT*, pages 199–218, 2009.

[19] Rishiraj Bhattacharyya, Avradip Mandal, and Mridul Nandi. **Security Analysis of the Mode of JH Hash Function**. In *FSE*, pages 168–191, 2010.

[20] Eli Biham and Orr Dunkelman. **A Framework for Iterative Hash Functions - HAIFA.** Cryptology ePrint Archive, Report 2007/278, 2007. `http://eprint.iacr.org/`.

[21] John Black, Phillip Rogaway, and Thomas Shrimpton. **Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV**. In *CRYPTO*, pages 320–335, 2002.

[22] Manuel Blum and Silvio Micali. **How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits**. *SIAM J. Comput.*, 13(4):850–864, 1984.

[23] Alexandra Boldyreva and Marc Fischlin. **Analysis of Random Oracle Instantiation Scenarios for OAEP and Other Practical Schemes**. In *CRYPTO*, pages 412–429, 2005.

[24] Alexandra Boldyreva and Marc Fischlin. **On the Security of OAEP**. In *ASIACRYPT*, pages 210–225, 2006.

[25] Alexandra Boldyreva, Hideki Imai, and Kazukuni Kobara. **How to Strengthen the Security of RSA-OAEP**. *IEEE Transactions on Information Theory*, 56(11):5876–5886, 2010.

[26] Dan Boneh. **Simplified OAEP for the RSA and Rabin Functions**. In *CRYPTO*, pages 275–291, 2001.

[27] Dan Boneh, Periklis A. Papakonstantinou, Charles Rackoff, Yevgeniy Vahlis, and Brent Waters. **On the Impossibility of Basing Identity Based Encryption on Trapdoor Permutations**. In *FOCS*, pages 283–292, 2008.

[28] Ran Canetti. **Towards Realizing Random Oracles: Hash Functions That Hide All Partial Information**. In *CRYPTO*, pages 455–469, 1997.

[29] Ran Canetti, Yevgeniy Dodis, Shai Halevi, Eyal Kushilevitz, and Amit Sahai. **Exposure-Resilient Functions and All-or-Nothing Transforms**. In *EUROCRYPT*, pages 453–469, 2000.

[30] Ran Canetti, Oded Goldreich, and Shai Halevi. **The random oracle methodology, revisited**. *J. ACM*, 51(4):557–594, 2004.

[31] Donghoon Chang, Sangjin Lee, Mridul Nandi, and Moti Yung. **Indifferentiable Security Analysis of Popular Hash Functions with Prefix-Free Padding**. In *ASIACRYPT*, pages 283–298, 2006.

[32] Donghoon Chang and Mridul Nandi. **Improved Indifferentiability Security Analysis of chopMD Hash Function**. In *FSE*, pages 429–443, 2008.

[33] Jean-Sébastien Coron. **On the Exact Security of Full Domain Hash**. In *CRYPTO*, pages 229–235, 2000.

[34] Jean-Sébastien Coron. **Optimal Security Proofs for PSS and Other Signature Schemes**. In *EUROCRYPT*, pages 272–287, 2002.

[35] Jean-Sébastien Coron. **Security Proof for Partial-Domain Hash Signature Schemes**. In *CRYPTO*, pages 613–626, 2002.

[36] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. **Merkle-Damgård Revisited: How to Construct a Hash Function**. In *CRYPTO*, pages 430–448, 2005.

[37] Jean-Sébastien Coron and Avradip Mandal. **PSS Is Secure against Random Fault Attacks**. In *ASIACRYPT*, pages 653–666, 2009.

[38] Jean-Sébastien Coron, David Naccache, and Julien P. Stern. **On the Security of RSA Padding**. In *CRYPTO*, pages 1–18, 1999.

[39] Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin. **The Random Oracle Model and the Ideal Cipher Model Are Equivalent**. In *CRYPTO*, pages 1–20, 2008.

[40] Giovanni Di Crescenzo, Richard J. Lipton, and Shabsi Walfish. **Perfectly Secure Password Protocols in the Bounded Retrieval Model.** In *TCC*, pages 225–244, 2006.

[41] Ivan Damgård. **A Design Principle for Hash Functions**. In *CRYPTO*, pages 416–427, 1989.

[42] Yevgeniy Dodis, Iftach Haitner, and Aris Tentes. **On the (In)Security of RSA Signatures**. Cryptology ePrint Archive, Report 2011/087, 2011. `http://eprint.iacr.org/`.

[43] Yevgeniy Dodis, Roberto Oliveira, and Krzysztof Pietrzak. **On the Generic Insecurity of the Full Domain Hash**. In *CRYPTO*, pages 449–466, 2005.

[44] Yevgeniy Dodis, Leonid Reyzin, Ronald L. Rivest, and Emily Shen. **Indifferentiability of Permutation-Based Compression Functions and Tree-Based Modes of Operation, with Applications to MD6**. In *FSE*, pages 104–121, 2009.

[45] Yevgeniy Dodis, Thomas Ristenpart, and Thomas Shrimpton. **Salvaging Merkle-Damgård for Practical Applications**. In *EUROCRYPT*, pages 371–388, 2009.

[46] Stefan Dziembowski. **Intrusion-Resilience Via the Bounded-Storage Model.** In *TCC*, pages 207–224, 2006.

[47] Amos Fiat and Adi Shamir. **How to Prove Yourself: Practical Solutions to Identification and Signature Problems**. In *CRYPTO*, pages 186–194, 1986.

[48] Marc Fischlin. **On the Impossibility of Constructing Non-interactive Statistically-Secret Protocols from Any Trapdoor One-Way Function**. In *CT-RSA*, pages 79–95, 2002.

[49] Marc Fischlin, Anja Lehmann, Thomas Ristenpart, Thomas Shrimpton, Martijn Stam, and Stefano Tessaro. **Random Oracles with(out) Programmability**. In *ASIACRYPT*, pages 303–320, 2010.

[50] Marc Fischlin and Dominique Schröder. **On the Impossibility of Three-Move Blind Signature Schemes**. In *EUROCRYPT*, pages 197–215, 2010.

[51] Ewan Fleischmann, Michael Gorski, and Stefan Lucks. **Some Observations on Indifferentiability**. In *ACISP*, pages 117–134, 2010.

[52] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. **RSA-OAEP Is Secure under the RSA Assumption**. *J. Cryptology*, 17(2):81–104, 2004.

[53] Rosario Gennaro, Yael Gertner, and Jonathan Katz. **Lower bounds on the efficiency of encryption and digital signature schemes**. In *STOC*, pages 417–425, 2003.

[54] Rosario Gennaro, Yael Gertner, Jonathan Katz, and Luca Trevisan. **Bounds on the Efficiency of Generic Cryptographic Constructions**. *SIAM J. Comput.*, 35(1):217–246, 2005.

[55] Rosario Gennaro and Luca Trevisan. **Lower Bounds on the Efficiency of Generic Cryptographic Constructions**. In *FOCS*, pages 305–313, 2000.

[56] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. **The Relationship between Public Key Encryption and Oblivious Transfer**. In *FOCS*, pages 325–335, 2000.

[57] Yael Gertner, Tal Malkin, and Steven Myers. **Towards a Separation of Semantic and CCA Security for Public Key Encryption**. In *TCC*, pages 434–455, 2007.

[58] Yael Gertner, Tal Malkin, and Omer Reingold. **On the Impossibility of Basing Trapdoor Functions on Trapdoor Predicates**. In *FOCS*, pages 126–135, 2001.

[59] O. Goldreich. **The Foundations of Cryptography-Volume 1**. Cambridge University Press, 2001.

[60] O. Goldreich. **The Foundations of Cryptography-Volume 2**. Cambridge University Press, 2004.

[61] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. **How to construct random functions**. *J. ACM*, 33(4):792–807, 1986.

[62] Oded Goldreich, Silvio Micali, and Avi Wigderson. **Proofs that Yield Nothing But Their Validity for All Languages in NP Have Zero-Knowledge Proof Systems**. *J. ACM*, 38(3):691–729, 1991.

[63] Shafi Goldwasser and Yael Tauman Kalai. **On the (In)security of the Fiat-Shamir Paradigm**. In *FOCS*, pages 102–, 2003.

[64] IETF-TLS Working Group. **Transport Layer Security**. http://www.ietf.org/html.charters/tls-charter.html, 2005.

[65] Shai Halevi, Steven Myers, and Charles Rackoff. **On Seed-Incompressible Functions**. In *TCC*, pages 19–36, 2008.

[66] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. **A Pseudorandom Generator from any One-way Function**. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[67] Shoichi Hirose. **Secure Block Ciphers Are Not Sufficient for One-Way Hash Functions in the Preneel-Govaerts-Vandewalle Model**. In *Selected Areas in Cryptography*, pages 339–352, 2002.

[68] Shoichi Hirose. **Provably Secure Double-Block-Length Hash Functions in a Black-Box Model**. In *ICISC*, pages 330–342, 2004.

[69] Shoichi Hirose. **Some Plausible Constructions of Double-block-length Hash Functions.** In *FSE*, pages 210–225, 2006.

[70] Shoichi Hirose, Je Hong Park, and Aaram Yun. **A Simple Variant of the Merkle-Damgård Scheme with a Permutation**. In *ASIACRYPT*, pages 113–129, 2007.

[71] Chun-Yuan Hsiao and Leonid Reyzin. **Finding Collisions on a Public Road, or Do Secure Hash Functions Need Secret Coins?** In *CRYPTO*, pages 92–105, 2004.

[72] Russell Impagliazzo and Michael Luby. **One-way Functions are Essential for Complexity Based Cryptography (Extended Abstract)**. In *FOCS*, pages 230–235, 1989.

[73] Russell Impagliazzo and Steven Rudich. **Limits on the Provable Consequences of One-Way Permutations**. In *STOC*, pages 44–61, 1989.

[74] RSA Security Inc. **PKCS #1 v2.1: RSA cryptography standard**. ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf, 2002.

[75] Jonathan Katz and Yehuda Lindell. **Introduction to Modern Cryptography**. Chapman & Hall/CRC, 2007.

[76] Jonathan Katz and Arkady Yerukhimovich. **On Black-Box Constructions of Predicate Encryption from Trapdoor Permutations**. In *ASIACRYPT*, pages 197–213, 2009.

[77] Eike Kiltz, Adam O'Neill, and Adam Smith. **Instantiability of RSA-OAEP under Chosen-Plaintext Attack**. In *CRYPTO*, pages 295–313, 2010.

[78] Eike Kiltz and Krzysztof Pietrzak. **On the Security of Padding-Based Encryption Schemes - or - Why We Cannot Prove OAEP Secure in the Standard Model**. In *EUROCRYPT*, pages 389–406, 2009.

[79] Jeong Han Kim, Daniel R. Simon, and Prasad Tetali. **Limits on the Efficiency of One-Way Permutation-Based Hash Functions**. In *FOCS*, pages 535–542, 1999.

[80] Kazukuni Kobara and Hideki Imai. **OAEP++ : A Very Simple Way to Apply OAEP to Deterministic OW-CPA Primitives**. Cryptology ePrint Archive, Report 2002/130, 2002. `http://eprint.iacr.org/`.

[81] Michael Luby. **Pseudorandomness and Cryptographic Applications**. Princeton University Press, 1996.

[82] Michael Luby and Charles Rackoff. **How to Construct Pseudorandom Permutations from Pseudorandom Functions**. *SIAM J. Comput.*, 17(2):373–386, 1988.

[83] Stefan Lucks. **A Failure-Friendly Design Principle for Hash Functions**. In *ASIACRYPT*, pages 474–494, 2005.

[84] m w.com. **Merriam-Webster's Open Dictionary**, 2011.

[85] Ueli M. Maurer. **Indistinguishability of Random Systems**. In *EUROCRYPT*, pages 110–132, 2002.

[86] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. **Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology**. In *TCC*, pages 21–39, 2004.

[87] Ralph C. Merkle. **One Way Hash Functions and DES**. In *CRYPTO*, pages 428–446, 1989.

[88] Steven Myers and Abhi Shelat. **Bit Encryption Is Complete**. In *FOCS*, pages 607–616, 2009.

[89] Mridul Nandi. **A Simple and Unified Method of Proving Indistinguishability**. In *INDOCRYPT*, pages 317–334, 2006.

[90] Moni Naor and Moti Yung. **Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks**. In *STOC*, pages 427–437, 1990.

[91] Jesper Buus Nielsen. **Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case**. In *CRYPTO*, pages 111–126, 2002.

[92] Tatsuaki Okamoto and David Pointcheval. **REACT: Rapid Enhanced-Security Asymmetric Cryptosystem Transform**. In *CT-RSA*, pages 159–175, 2001.

[93] Pascal Paillier. **Impossibility Proofs for RSA Signatures in the Standard Model**. In *CT-RSA*, pages 31–48, 2007.

[94] Chris Peikert and Brent Waters. **Lossy Trapdoor Functions and their Applications**. In *STOC*, pages 187–196, 2008.

[95] Duong Hieu Phan and David Pointcheval. **Chosen-Ciphertext Security without Redundancy**. In *ASIACRYPT*, pages 1–18, 2003.

[96] Bart Preneel, René Govaerts, and Joos Vandewalle. **Hash Functions Based on Block Ciphers: A Synthetic Approach**. In *CRYPTO*, pages 368–378, 1993.

[97] Charles Rackoff and Daniel R. Simon. **Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack**. In *CRYPTO*, pages 433–444, 1991.

[98] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. **Notions of Reducibility between Cryptographic Primitives**. In *TCC*, pages 1–20, 2004.

[99] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. **Careful with Composition: Limitations of the Indifferentiability Framework**. In *EUROCRYPT*, pages 487–506, 2011.

[100] Thomas Ristenpart and Thomas Shrimpton. **How to Build a Hash Function from Any Collision-Resistant Function**. In *ASIACRYPT*, pages 147–163, 2007.

[101] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. **A Method for Obtaining Digital Signatures and Public-Key Cryptosystems**. *Commun. ACM*, 21(2):120–126, 1978.

[102] Phillip Rogaway and Thomas Shrimpton. **Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance**. In *FSE*, pages 371–388, 2004.

[103] John Rompel. **One-Way Functions are Necessary and Sufficient for Secure Signatures**. In *STOC*, pages 387–394, 1990.

[104] Steven Rudich. **Limits on the Provable Consequences of One-way Functions**. PhD thesis, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel, 2004.

[105] Victor Shoup. **OAEP Reconsidered**. *J. Cryptology*, 15(4):223–249, 2002.

[106] Thomas Shrimpton and Martijn Stam. **Building a Collision-Resistant Compression Function from Non-compressing Primitives**. In *ICALP (2)*, pages 643–654, 2008.

[107] Daniel R. Simon. **Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions?** In *EUROCRYPT*, pages 334–345, 1998.

[108] Martijn Stam. **Beyond Uniformity: Better Security/Efficiency Tradeoffs for Compression Functions**. In *CRYPTO*, pages 397–412, 2008.

[109] Martijn Stam. **Blockcipher-Based Hashing Revisited**. In *FSE*, pages 67–83, 2009.

[110] Ronald L. Rivest (Principal Submitter). **The MD6 hash function – A Proposal to NIST for SHA-3**. Submission to NIST, 2008.

[111] Yevgeniy Vahlis. **Two Is a Crowd? A Black-Box Separation of One-Wayness and Security under Correlated Inputs**. In *TCC*, pages 165–182, 2010.

[112] Hoeteck Wee. **One-Way Permutations, Interactive Hashing and Statistically Hiding Commitments**. In *TCC*, pages 419–433, 2007.

[113] Hongjun Wu. **The Hash Function JH**. Submission to NIST (Round 1/2), 2009.

[114] Hongjun Wu. **The Hash Function JH: Round 3**. Submission to NIST (round 3), 2011.

[115] Andrew Chi-Chih Yao. **Theory and Applications of Trapdoor Functions (Extended Abstract)**. In *FOCS*, pages 80–91, 1982.