UNIVERSITÉ DU LUXEMBOURG

PhD-FSTC-2012-18
The Faculty of Sciences, Technology and Communication

DISSERTATION

Presented on 25/06/2012 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE

by

Avradip Mandal
Born on 29 August 1982 in Kolkata (India)

Provable Security and Indifferentiability

Dissertation defense committee

Dr. Alex Biryukov, Chairman
Associate Professor, Université du Luxembourg

Dr. Jean-Sébastien Coron, dissertation supervisor
Associate Professor, Université du Luxembourg

Dr. Yevgeniy Dodis
Associate Professor, New York University

Dr. Volker Müller, Vice Chairman
Associate Professor, Université du Luxembourg

Dr. David Pointcheval
Senior Researcher, CNRS, Head of the Crypto Team at ENS/INRIA CASCADE Project

Abstract

In this thesis we consider different problems related to provable security and indifferentiability framework. Ideal primitives such as random oracles, ideal ciphers are theoretical abstractions of cryptographic hash functions and block ciphers respectively. These idealized models help us to argue security guarantee for various cryptographic schemes, for which standard model security proofs are not known. In the first part of this thesis we consider the problems related to ideal primitive construction starting from a different ideal primitive. We adopt the indifferentiability framework proposed by Maurer et. al. in TCC'04 for this purpose. The indifferentiability framework helps us to preserve the security guarantee of cryptographic schemes when the ideal primitives are replaced by indifferentiable constructions, even when the ideal primitives are used in a public manner.

At first, we consider the problem of ideal cipher domain extension. We show the 3-round Feistel construction, built using n-bit ideal ciphers are actually indifferentiable from a $2n$-bit ideal cipher. We also consider other related issues such as, why 2-round Feistel is not sufficient, security analysis in standard indistinguishability model for both 2 and 3 round constructions, etc. Afterwards, we consider the open problem: whether 6-round Feistel construction using random round functions is indifferentiable from a random invertible permutation or not. We give a partial positive answer to this question. We show the construction is actually publicly-indifferentiable (which is a restricted version of full indifferentiability) from an invertible random permutation.

In the later part of the thesis, we concentrate on some issues related to the security of Probabilistic Signature Scheme (PSS). PSS with RSA trapdoor is a widely deployed randomized signature scheme. It is known to be secure in Random Oracle model. However, recently randomized signature scheme such as ISO/IEC 9796-2 is shown to be susceptible to hardware fault attacks. In this work we show, PSS is actually secure against random fault attacks in random oracle model. Afterwards, we consider the open problem related to standard model security of PSS. We give a general negative result in this direction. We rule out existence of any black box proof technique showing security of PSS in standard model.

Acknowledgements

This doctoral thesis is not only the result of my own endeavor, but guidance, inspiration and assistance of many people. I am deeply grateful to all of them.

This thesis would not have been possible without the active guidance and advice of my advisor Jean-Sébastien Coron. He set an excellent example as a world class researcher and supervisor, gave me academic freedom and encouraged me for independent exploration. For the past three and half years, I profoundly enjoyed the discussions with him, about various open problems in the areas of cryptology. His valuable insights always guided me to have a better understanding of the problem, and many times toward a solution.

I am deeply grateful to Volker Müller and David Pointcheval for being doctoral committee members, to Yevgeniy Dodis for serving as thesis jury and to Alex Biryukov for chairing the jury.

I will always be indebted to Alfred Menezes and Mridul Nandi for introducing me to the exciting world of Cryptography during my masters studies at University of Waterloo. I greatly appreciate effective collaborations and fruitful discussions with my external co-authors: Antoine Joux, David Naccache, Valérie Nachef, Jacques Patarin, Yannick Seurin and Mehdi Tibouchi. I am specially thankful to Rishiraj Bhattacharyya - my long time friend from school days, with whom I did numerous collaborative works during the past few years.

My deepest gratitude goes to Ranoj Banik, Prabir Banerjee, my mathematics teachers in middle school days and Sushmita Banik, my childhood tutor.

My many thanks go to the colleagues at the Laboratory of Algorithmics, Cryptology and Security of the University of Luxembourg for the superb working environment. I would like to thank Jean-François Gallais with whom I shared the office for the last three and half years, for the countless scientific (and non-scientific) discussions we had. It was a pleasure to be on the team with David Galindo, Johann Großschädl, Dmitry Khovratovich, Gaëtan Leurent, Zhe Liu, Ivica Nikolić, Arnab Roy, Deike Priemuth-Schmid, Praveen Kumar Vadnala, Srinivas Vivek Venkatesh, Ralf-Philipp Weinmann and Bin Zhang. I appreciate the help of LACS secretaries Ragga Eyjolfsdottir, Mireille Kies and Fabienne Schmitz with administrative tasks and paperwork.

I am thankful to all my friends in Luxembourg, with whom I shared a wonderful time.

My deepest appreciation goes to Dia, for her love, patience and support. Lastly, I would like to thank my parents and my brother Avikarsha, for their unconditional support and encouragement.

Avradip Mandal, June 2012

Contents

List of Figures

Chapter 1

Introduction

"Cryptography is concerned with the conceptualization, definition, and construction of computing systems that address security concerns ."
– 'Foundations of Cryptography' by Oded Goldreich, 1997

Contents

1.1 Cryptography

Cryptography is the science of securing communications. The fundamental and classical task of cryptography is to provide confidentiality by encryption methods. The message to be transmitted – it can be text, numerical data, an executable program or any other kind of information – is called plaintext. Alice encrypts the plaintext m and obtains ciphertext c. The ciphertext c is transmitted to Bob. Bob turns the ciphertext back into the plaintext by decryption. To decrypt, Bob needs some secret information, a secret decryption key. The adversary Eve still may intercept the ciphertext. However, the encryption should guarantee secrecy and prevent Eve from deriving any information about the plaintext from the observed ciphertext.

Encryption is very old. For example, the Caesar's shift cipher, where each plaintext character is replaced by the character 3 to the right modulo 26, (i.e. a is replaced by d, b by e,..., y by b, and z by c) is more than 2000 years ago. Every encryption method provides an encryption algorithm E and a decryption algorithm D. In symmetric-key cryptography both algorithms depend on the same secret key k. Caesar's cipher is an example of symmetric-key encryption scheme where the key is the offset 3. The Data Encryption Standard (DES) is another example of symmetric-key encryption.

In 1976 W. Diffie and M.E. Hellman published their famous paper, *New directions in Cryptography* [DH76]. There they introduced the revolutionary concept of public-key cryptography. They also provided a solution to the long standing problem of key exchange and pointed the way to digital signatures. *Public-key encryption* methods are asymmetric. Each recipient of messages has his personal key $k = (pk, sk)$, consisting of two parts: pk is the encryption key and is made public, sk is the decryption key and is kept secret. If Alice wants to send a message m to Bob, she encrypts m by use of Bob's publicly known encryption key pk. Bob decrypts the ciphertext by use of his decryption key sk, which is known only to him.

Mathematically speaking public-key encryption is a so-called *one-way function* with a *trapdoor*. Anyone can easily encrypt a plaintext using the public key pk, but the other direction is difficult. It is practically impossible to deduce the plaintext from the ciphertext, without knowing the secret key sk (which is called the trapdoor information).

Public-key encryption methods require more complex computations and are less efficient than classical symmetric methods. Thus symmetric methods are used for the encryption of large amounts of data. Before applying symmetric encryption, Alice and Bob have to agree on a key. To keep this key secret, they need a secure communication channel. It is common practice to use public-key encryption for this purpose.

The objectives of cryptography

Providing confidentiality is not the only objective of cryptography. Cryptography is also used to provide solutions to other problems.

1. **Data integrity**: The receiver of a message should be able to check whether the message was modified during transmission, either accidentally or deliberately. No one should be able to substitute a false message for the original message, or for part of it.

2. **Authentication**: The receiver of a message should be able to verify its origin. No one should be able to send a message to Bob and pretend to be Alice (*data origin authentication*). When initiating a communication, Alice and Bob should be able to identify each other (*entity authentication*).

3. **Non-repudiation**: The sender should not be able to later deny that she sent a message.

If messages are written on paper, the medium – paper – provides security against manipulation. Handwritten personal signatures are intended to guarantee authentication and non-repudiation. If electronic media are used, the medium itself provides no security at all, since it is easy to replace some bytes in a message during its transmission over a computer network, and it is particularly easy if the network is publicly accessible, like the Internet.

So, while encryption has a long history, the need for techniques providing data integrity and authentication resulted from the rapidly increasing significance of electronic communication.

There are symmetric as well as public-key methods to ensure the integrity of messages. *Digital signatures* require public-key methods. As with classical handwritten signatures, they are intended to provide authentication and non-repudiation. Note that non-repudiation is an indispensable feature if digital signatures are used to sign contracts. Digital signatures depend on the secret key of the signer – they can generated only by him. On the other hand, anyone can check whether a signature is valid, by a publicly known verification algorithm *Verify*, which depends on the public key of the signer. It is common not to sign the message itself, but to apply a *cryptographic hash function* first and then sign the hash value. Digital Signatures depend on the message. Different messages generate different signatures. So they can also be used to provide message authentication. The symmetric-key method to ensure integrity of messages is achieved by *Message Authentication Codes* (MAC).

The primary goal of cryptography is to keep the plaintext secret from eavesdroppers trying to get some information about the plaintext. As discussed before, adversaries may also be active and try to modify the message. Then cryptography is expected to guarantee the integrity of messages. Adversaries are assumed to have complete access to the communication channel.

Cryptanalysis is the science of studying attacks against cryptographic schemes. Successful attacks may, for example, recover the plaintext (or parts of the plaintext) from the ciphertext, substitute parts of the original message, or forge digital signatures. Cryptography and cryptanalysis are often subsumed by the more general term cryptology.

A fundamental assumption in cryptanalysis was first stated by A. Kerkhoff in the nineteenth century, and is usually referred to as Kerkhoff's principle. It states that the adversary knows all the details of the cryptosystem, including algorithms and their implementation. According to this principle, the security of a cryptosystem must be entirely based on the secret keys.

Attacks against Encryption Schemes

Attacks on the secrecy of an encryption scheme try to recover plaintexts from ciphertexts, or even more drastically the secret key. In the following we only consider a passive attacker Eve, who does not try to modify the messages. However the attacker has access to plaintexts and ciphertexts, and she may have control over choosing plaintexts and/or ciphertexts. Of course she does not have access to the secret key. The possible attacks depend on the actual resources of Eve. They are usually classified as follows:

- **Ciphertext-only attack**: Eve has the ability to obtain ciphertexts. This is likely to be the case in any encryption scenario. Even if Eve cannot perform other more sophisticated attacks, one must assume that she can get access to the encrypted messages. An encryption method that cannot resist a ciphertext-only attack is completely insecure.

- **Known-plaintext attack**: Eve has the ability to obtain plaintext-ciphertext pairs. Using the information from these pairs, she attempts to decrypt a ciphertext for which she does not have the plaintext.
- **Chosen-plaintext attack**: Eve has the ability to obtain ciphertexts for plaintexts of her choosing. Then she attempts to decrypt a ciphertext for which she does not have the plaintext. Here she has access to the encrypting device only once. This means after she starts analysis, she cannot access the encrypting device any more.
- **Adaptively-chosen-plaintext attack**: This is the same as the previous attack, except now Eve may do some analysis on the plaintext-ciphertext pairs, and subsequently, get more pairs. She may switch between gathering pairs and performing the analysis as often as she likes. This means that she has either lengthy access to the encrypting device or can somehow make repeated use of it.
- **Chosen-ciphertext and adaptively-chosen ciphertext attack**: These two attacks are similar to the above plaintext attacks. Eve can choose ciphertexts and gets the corresponding plaintexts. She has access to the decryption device.

Attacks against Signature Schemes

Attacking a signature scheme usually implies signature forging. Forging means one has to generate a valid signature for a message of her choice without access to the secret key. Like encryption schemes, an attacker can mount various types of attacks depending on the resources she has. Below we describe various kind of attack models:

- **Zero message attack**: This is the strongest attack model possible. The attacker Eve has only access to the public key of the signature scheme. Just from this information she tries to forge a signature for a message of her choice.
- **Chosen message attack**: The attacker Eve has the ability to obtain signatures for the messages of her choosing. She has access to a signing oracle. However, she needs to submit all her queries to the oracle at once, right at the beginning. afterwards, she does not have access to the oracle anymore. In the end, Eve is supposed forge a valid signature for a message of her choice for which she did not make query to the signing oracle. In a variant of this attack model, Eve is allowed to forge signatures for messages which she already queried to the signing oracle; as long as she outputs a different (message, signature) pair from the ones she received.
- **Adaptively chosen message attack**: This is the standard attack model for signature schemes. This is similar to the previous attack. However, here the attacker Eve is more powerful. She can use the signature oracle adaptively by making queries based on the previous answers she received. In the end

Eve is supposed to produce a forged signature for a message of her choice, for which she did not make a query to the signing oracle. As before, one can also consider a relaxed variant of this attack model.

Fault Attacks

In real world, an adversary can go beyond the theoretical attack models and attack the implementation rather than the specification. For example in a smart card implementation of a signature scheme, during a signature evaluation an attacker can gather lots of side channel information such as time required to complete the signature evaluation, power consumption profile, electro-magnetic radiation profile, etc. All these data, leak some information about the secret key. Using these information an attacker becomes more powerful and might be able to break cryptographic schemes which are secure against conventional attacks. Moreover, the scope of a real world attacker is not only limited to passive collection of side channel information. It can actively introduce some temporary hardware malfunction forcing the implementation to release more information about the secret key. These kind of attacks are called fault injection attacks or *fault attacks*, as they inject hardware faults (usually by means of short electro-magnetic impulses or by introducing some spikes in the power source) during some computation involving the secret key. For some cryptographic schemes fault attacks can be highly effective, completely breaking the security of the scheme by recovering the secret key. We will see more about fault attacks in Chapter 5.

1.2 Provable security

It is desirable to design cryptosystems that are provably secure. Provably secure means mathematical proofs show that the cryptosystem resists certain types of attacks. Pioneering work in this field was done by C.E. Shannon. In his information theory, he developed a measurement for the amount of information associated with a message and the notion of perfect secrecy. A *perfectly secret* cipher perfectly resists all ciphertext-only attacks. An adversary gets no information whatsoever about the plaintext, even if his resources in computational power and time are unlimited. *Vernam's one-time pad* which encrypts a message m by XORing it bitwise with a truly random bit string, is the most famous perfectly secret cipher. It even resists all the passive attacks mentioned. This can be mathematically proven by Shannon's theory. Unfortunately Vernam's one-time pad and all perfectly secret ciphers are usually impractical. It is not practical in most situations to generate and handle truly random bit sequences of sufficient length as required for perfect secrecy.

More recent approaches to provable security therefore abandon the ideal of perfect secrecy and the unrealistic assumption of unbounded computing power of adversary. Only attacks that might be *feasible* are taken into account. Feasible means that the attacks can be performed by an *efficient algorithm*. Certainly attacker algorithms with non-polynomial running times are not efficient. Conversely algorithms with polynomial running times are often considered efficient ones. If the

attacker uses probabilistic algorithms then average running times are taken into account.

The security of a public-key cryptosystem is based on the hardness of some computational problem. For example, the secret keys of an RSA scheme could be easily deduced if computing the factors of a large integer was possible. However, it is believed that factoring large integers is infeasible. There are no mathematical proofs for the hardness of the computational problems used in public-key systems. Therefore, security proofs for public-key methods are always conditional. They depend on the validity of underlying assumptions.

Random Oracle and Other idealized models

However, as it turns out the computational assumptions are not always enough to design efficient and secure cryptographic schemes. To overcome this problem Fiat and Shamir first proposed the idea of using Random Oracles in cryptographic protocols. A random oracle is a theoretical black box (ideal primitive) that responds to every query with a (truly) random response chosen uniformly from its output domain. For repeated queries the random oracle always outputs the same response. In their Crypto '86 paper [FS86], Fiat and Shamir showed how random oracles can be used to construct a signature scheme without any need of interactions. Afterwards, Bellare and Rogaway [BR93, BR96] proposed efficient signature schemes such as Full Domain Hash (FDH) and Probabilistic Signature Scheme (PSS), as well as efficient public key encryption schemes such as OAEP [BR94] which are secure in the random oracle model. They used random oracles as a public primitive which are available to all parties involved in the scheme including the attacker. Similar to the random oracle model we also have other idealized models such as *ideal cipher model* (an idealized version of a block cipher), *random permutation model*, etc.

Bellare and Rogaway suggested that for practical purposes random oracles can be replaced by some well known cryptographic hash function. However, mathematically speaking a public hash function can never be a random oracle. Hash functions do not use any random coins, for a fixed input it always gives fixed output. That is why Canetti et. al. [CGH98, CGH04] argued a security proof in the random oracle model is heuristic at best. In fact, they showed existence of some contrived cryptographic schemes which are secure in the random oracle model. However, the security guarantee completely vanishes whenever the random oracle is replaced by some real hash function. Hence, from a provable security point of view security proofs in so called *standard model* (here one assumes some well studied security property of cryptographic hash functions such as collision resistance, pre-image resistance, etc.) are always desirable. However, in many cases the standard model schemes turn out to be inefficient, ruling out their practical applicability. As a result, till date random oracle remains popular in cryptographic community, after all a security proof in the random oracle model is better than no proof at all.

Equivalence of idealized models and indifferentiability

Following Bellare and Rogaway's seminal work, many cryptographic schemes were presented with a proof of security in some idealized model such as the random oracle model, the ideal cipher model, the random permutation model, etc. Moreover, they also required different domain and range depending on the application. For example, for a generalized signature scheme without any bound on the message space we require a variable input length (VIL) hash function which behaves as a random oracle; whereas, for some specialized schemes a finite input length (FIL) hash function might be sufficient.

So a natural question arises, how we should design such hash functions where the classical security requirements such as collision resistance or pre-image resistance do not suffice. With the advent of random oracle model cryptographic schemes, the new security requirement is: the hash functions should behave as a random oracle (albeit heuristically). Also one might ponder, whether there are any relations between various idealized models. Would it be possible to implement one ideal primitive with another ideal primitive, while preserving the security guarantee.

Maurer et. al. in their TCC '04 paper [MRH04] introduced the notion of indifferentiability, which extends the well known indistinguishability framework to public primitives. The indifferentiability framework actually shows the way to answer our second question in previous paragraph. As we will see in Chapter 2, it is an indispensable tool to show equivalence between two public ideal primitives. Following Maurer et. al.'s work Coron et. al. in Crypto '05 [CDMP05] showed how one should design a hash function which will behave as a variable input length random oracle. They showed, some restricted version of well known hash function design methods such Merkle-Damgard mode of operation and Davis-Mayer mode of operation are actually sufficient. The Merkle-Damgard mode of operation uses a fixed input length function as building block, where as the Davis-Mayer mode of operation uses a block cipher as a building block. Coron et. al.'s Crypto '05 paper showed that the Merkle-Damgard mode of operation and Davis-Mayer mode of operation are actually good candidates to replace Random Oracles in cryptographic schemes, provided we are willing to assume that the fixed input length function and the block cipher behave as fixed input length random oracle and ideal cipher respectively.

However, we should remember that even if we use an indifferentiable hash function as suggested by Coron et. al., a security proof in the random oracle model still remains heuristic at best. An indifferentiable hash function only guarantees that there is nothing wrong with the selected mode of operation. The scheme can be still susceptible to an attack if there is any weakness in the underlying fixed input length function or the underlying block cipher. The indifferentiability framework helps us to simplify our heuristic security assumptions. Instead of assuming a complex hash function behaves as a variable input length random oracle we can assume a simpler to understand fixed input length function or block cipher behave as a random oracle or an ideal cipher respectively.

Black Box impossibility of reduction based security proofs

Cryptographic security proofs based on some computationally difficult problem usually follow a reductionist approach. At first we assume there exists some adversary which breaks the cryptographic scheme. Then using the initial adversary as a *black box* we build a new adversary which can successfully solve the computationally difficult problem whenever the initial adversary is successful. However, as we believe the computational problem is hard to solve, by contradiction we deduce the cryptographic scheme is secure. This kind of black box reduction technique is immensely popular in present day provable security literature. However, not all cryptosystems can be proven secure in all security models. Not only that, there exist cryptosystems for which neither we can find a security proof, nor we can show a concrete attack to prove the cryptosystem is insecure. A typical example of this are signature schemes such as Full Domain Hash (FDH), Probabilistic Signature Scheme (PSS). Even though they were proven secure in the random oracle model by Bellare and Rogaway [BR93, BR96], no standard model security proof is known to exist. However, a successful attack against some implementation of these signature schemes is actually equivalent to finding a weakness in some well studied hash function; which is hard.

Hsiao and Reyzin in their Crypto '04 [HR04b] paper proposed a technique which partially solves this dilemma. Instead of answering the question whether these cryptographic schemes are secure in the standard model, their technique helps us to show: it is impossible to find a black box based reduction strategy to prove the security of FDH and PSS in the standard model. In Crypto'07 [DOP05], Dodis et. al. used this technique to argue generic insecurity of FDH in the standard model. Later, in Eurocrypt '09 [KP09], Kiltz and Pietrzak adapted the same technique for Optimal Asymmetric Encryption scheme (OAEP). In Chapter 6, we extend these generic insecurity results to PSS.

1.3 Organization

This thesis work is primarily based on four papers, previously published in Asiacrypt'09, TCC'10, PKC'11 and TCC'12 conference proceedings. In the next three chapters we present the results related to indifferentiability framework. In the last two chapters we will see some new security and insecurity results related to the probabilistic signature scheme (PSS).

- In **Chapter 2**, we give a formal introduction to the indifferentiability framework [MRH04, CDMP05] and the motivation behind its importance. We also introduce two other variants of indifferentiability notion, namely indifferentiability in honest but curious model [DP06] and public indifferentiability [DRS09].

- **Chapter 3** is based on the TCC'10 paper *A Domain Extender for the Ideal Cipher*, which is a joint work with Jean-Sébastien Coron, Yevgeniy Dodis and Yannick Seurin [CDMS10]. Over here we consider the problem of extending an

ideal cipher domain. We show the 3-round Feistel construction based on a n-bit ideal cipher is actually indifferentiable from a $2n$-bit ideal cipher. We also show 2-rounds are actually not enough by giving a simple attack. Moreover, we consider our construction in standard indistinguishability model for usage in secret key scenario. We show 2-rounds are actually enough to build a $2n$-bit tweakable block cipher from a n-bit tweakable block cipher. We also show with 3-rounds we can get a $2n$-bit tweakable block cipher which gives beyond birthday security guarantee.

- **Chapter 4** is based on the TCC'12 paper *On the Public Indifferentiability and Correlation Intractability of the 6-Round Feistel Construction*, which is a joint work with Jacques Patarin and Yannick Seurin [MPS12]. Here, we show that the Feistel construction with six rounds and random round functions is publicly indifferentiable from a random invertible permutation (a result not known to hold for full indifferentiability). To prove this result we introduce a new and simpler variant of indifferentiability called sequential indifferentiability and show this is equivalent to public indifferentiability for stateless ideal primitives. We then prove that the 6-round Feistel construction based on random round functions is sequential indifferentiable from a random invertible permutation. We also show sequential indifferentiability implies correlation intractability, a notion introduced by Canetti et. al. in their work on the limitations of the random oracle model [CGH98, CGH04]. We also show the 3-round domain extender construction from Chapter 3, actually provides beyond birthday security guarantee in sequential indifferentiability model.

- **Chapter 5** is based on the Asiacrypt'09 paper *PSS is Secure against Random Fault Attacks*, which is a joint work with Jean-Sébastien Coron [CM09]. In this chapter we consider the security of well known Probabilistic Signature Scheme (PSS) against fault attacks. A fault attack consists in inducing hardware malfunctions in order to recover secrets from electronic devices. One of the most famous fault attack is Bellcore's attack against RSA with Chinese Remainder Theorem (CRT) implementation. The fault attack applies to any deterministic RSA-CRT based signature scheme, for example FDH. Recently, the attack was extended to *randomized* encodings based on the ISO/IEC 9796-2 signature standard. Extending the attack to other randomized encodings remains an open problem. In this chapter we show Bellcore's attack can not be applied to PSS; namely we show that PSS is provably secure against random fault attacks in the random oracle model, assuming that inverting RSA is hard.

- **Chapter 6** is based on the PKC'11 paper *On the Impossibility of Instantiating PSS in the Standard Model*, which is a joint work with Rishiraj Bhattacharyya [BM11a]. PSS was proven to be secure in The Random Oracle Model by Bellare and Rogaway [BR96]. In Chapter 5 we even showed PSS resists random fault attacks in the random oracle model. In this chapter, we consider the problem of securely instantiating PSS in the standard model. Our main result

is a black-box impossibility result showing that one can not prove unforgeability of PSS against chosen message attacks using blackbox techniques, assuming existence of *ideal trapdoor permutations* [KP09] or *lossy trapdoor permutations* [PW08]. 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.

Chapter 2

Indifferentiability Framework

In this chapter we discuss the motivation behind indifferentiability framework, introduced by Maurer et. al. in TCC'04 [MRH04]. We will see what do we formally mean by indifferentiability and why this might be an important tool for hash function design.

Contents

2.1 Introduction

Indifferentiability is actually a generalization of indistinguishability. As explained in [Sho04], indistinguishability is an important concept for reduction based cryptographic security proofs. However, the indistinguishability alone is not always enough for the security proofs to go through; specially, in the public key settings. The following two simple examples taken from [MRH04] show when the concept of indistinguishability is useful, and why it is not always enough.

Example 2.1 Let $\mathcal{T}$ be a source of truly random bits (secret between the communicating parties), $\mathcal{S}$ be a pseudo-random bit generator (with secret key shared between the communicating parties). $\mathcal{C}^{(\cdot)}$ be the XOR based encryption. In other words, $\mathcal{C}^{\mathcal{T}}$ denotes the one time pad and $\mathcal{C}^{\mathcal{S}}$ denotes an additive stream cipher with key-stream generator $\mathcal{S}$. The security of $\mathcal{C}^{\mathcal{S}}$ follows from the security of $\mathcal{C}^{\mathcal{T}}$ and the fact that, for any efficient distinguisher (or adversary), $\mathcal{S}$ behaves essentially like $\mathcal{T}$, i.e., $\mathcal{S}$ and $\mathcal{T}$ are (computationally) indistinguishable.

Example 2.2 Let $\mathcal{T}$ be a random oracle $\mathcal{R}$, (i.e., a publicly accessible random function) and let $\mathcal{S}$ be a hash function $\mathcal{H}^{\mathcal{F}}$, where $\mathcal{H}$ is a hash algorithm depending

on a public component $\mathcal{F}$ (can be a block cipher with public key or an S-Box). In contrast to pseudo-randomness (where the component is secret), no hash function can implement a random oracle in the above sense, as proved by Canetti, Goldreich, and Halevi [CGH04]. In other words, there exists a cryptosystem $\mathcal{C}^{(\cdot)}$ such that $\mathcal{C}^{\mathcal{R}}$ is secure while $\mathcal{C}^{\mathcal{H}^{\mathcal{F}}}$ is insecure for any hash algorithm $\mathcal{H}$ and any public real life component $\mathcal{F}$. However, one crucial observation is if we are willing to model $\mathcal{F}$ as an public ideal component such as, fixed domain random function, random permutation, ideal cipher, etc; $\mathcal{C}^{\mathcal{H}^{\mathcal{F}}}$ might still be secure. Even though, the security of $\mathcal{C}^{\mathcal{R}}$ and indistinguishability of $\mathcal{H}^{\mathcal{F}}$ and $\mathcal{R}$ in secret key setting are not enough to guarantee such security.

Indistinguishability and Indifferentiability

Two systems $\mathcal{S}$ and $\mathcal{T}$ are said to be indistinguishable if no (efficient) algorithm $\mathcal{D}$ connected to either $\mathcal{S}$ or $\mathcal{T}$, is able to decide whether it is interacting with $\mathcal{S}$ or $\mathcal{T}$. The security of a cryptosystem $\mathcal{C}^{\mathcal{S}}$ involving a component $\mathcal{S}$ is typically proven by considering the cryptosystem $\mathcal{C}^{\mathcal{T}}$ (replacing $\mathcal{S}$ by an idealized component $\mathcal{T}$). As we have seen in Example 2.1 the original system $\mathcal{C}^{\mathcal{S}}$ is secure if,

1. the system $\mathcal{C}^{\mathcal{T}}$ is secure and
2. the component $\mathcal{S}$ is indistinguishable from the component $\mathcal{T}$.

The notion of reducibility and indistinguishability are in fact closely related. A system $\mathcal{U}$ is said to be reducible to system $\mathcal{V}$ if the system $\mathcal{V}$ can be used to construct a new system $\mathcal{B}^{\mathcal{V}}$ which is indistinguishable from $\mathcal{U}$. The notion of reducibility is really useful for cryptographic security proofs. If $\mathcal{U}$ is reducible to $\mathcal{V}$, then for any cryptosystem $\mathcal{C}^{\mathcal{U}}$ using $\mathcal{U}$ as a component, there is another cryptosystem $\mathcal{C}^{\mathcal{B}^{\mathcal{V}}}$ using $\mathcal{V}$ as a component, having the same functionality and the same security as $\mathcal{C}^{\mathcal{U}}$.

However, these security reductions are all subject to the assumption that the cryptosystem $\mathcal{C}$ retains exclusive rights to the components ($\mathcal{U}$ or $\mathcal{V}$). All other parties, including the possible adversary, are unable to directly influence the component's behavior. Only way they can interact with the components is via the cryptosystem $\mathcal{C}$. As explained in Example 2.2, this is not always the case. If we have a cryptosystem $\mathcal{C}^{\mathcal{R}}$ provably secure for a random oracle $\mathcal{R}$, the provable security always gets lost whenever we instantiate the random oracle $\mathcal{R}$ by a real life public hash function $\mathcal{H}$. Moreover, even if we consider an idealized hash function $\mathcal{H}^{\mathcal{F}}$ (based on an ideal but public component $\mathcal{F}$), the notion of indistinguishability does not really help us to infer anything about the security of the cryptosystem $\mathcal{C}^{\mathcal{H}^{\mathcal{F}}}$.

In order to extend the definition of indistinguishability such as to include these type of systems Maurer et. al. proposed the notion of indifferentiability [MRH04]. In particular, if a component $\mathcal{H}^{\mathcal{F}}$ based on an ideal component $\mathcal{F}$, is indifferentiable from a component $\mathcal{R}$; then the security of any[1] cryptosystem $\mathcal{C}^{\mathcal{R}}$ based on

[1]Recently, Ristenpart et. al. actually showed this is not true for *any* cryptosystem [RSS11]. However, indifferentiability framework still includes a large number of present day cryptosystems.

ideal component $\mathcal{R}$ implies the security of the cryptosystem $\mathcal{C}^{\mathcal{H}^{\mathcal{F}}}$ based on ideal component $\mathcal{F}$.

The next section describes the formal definition of indifferentiability. We will also go through a short proof of our previous claim, i.e. indifferentiability extends the notion of indistinguishability from a secret world to the public world.

2.2 Indifferentiability

Before going to the formal definition of indifferentiability, let us see what do we formally mean by ideal primitives. We define *ideal primitive* as a probabilistic algorithmic entity which receives inputs from one of the parties and deliver its output immediately to the querying party. *Random oracles*, *random permutations* and *ideal ciphers* are three ideal primitives that we will encounter through out this work.

A *random oracle* is an ideal primitive which provides a random output for each new query. Identical input queries are given the same answer. By random oracle we normally imply a variable input length random oracle $RO : \{0,1\}^* \to \{0,1\}^n$. We can also have a fixed input length random oracle $RO : \{0,1\}^k \to \{0,1\}^n$.

A *random permutation* is an ideal primitive which models a randomly chosen permutation E over $\{0,1\}^n$. This is similar to a fixed input length random oracle $RO : \{0,1\}^n \to \{0,1\}^n$, with the added restriction - different input queries must provide different answers. Moreover, it provides oracle access to both E and E^{-1}; that is on query $(0,m)$ the primitive answers $c = E(m)$, and on query $(1,c)$ the primitive answers m such that $c = E(m)$.

An *ideal cipher* is an ideal primitive which models a randomly chosen block-cipher $E : \{0,1\}^k \times \{0,1\}^n$. Each key $k \in \{0,1\}^k$ defines a random permutation $E_k = E(k,\cdot)$ on $\{0,1\}^n$. The ideal cipher provides to E and E^{-1}; that is on query $(0,k,m)$, the primitive answers $c = E_k(m)$, and on query $(1,k,c)$, the primitive answers m such that $c = E_k(m)$.

The indifferentiability notion in [MRH04] is given in the framework of random systems providing interfaces to other systems. However following Coron et. al. [CDMP05], here we consider the equivalent notion of indifferentiability for interactive Turing Machines.

Let F_i, G_i be probabilistic oracle algorithms. We define the *advantage* of the adversary $\mathcal{A}$ at distinguishing (F_1, F_2) from (G_1, G_2) as

$$\mathsf{Adv}_{\mathcal{A}}((F_1,F_2),(G_1,G_2)) = |\Pr[\mathcal{A}^{F_1,F_2} = 1] - \Pr[\mathcal{A}^{G_1,G_2} = 1]|.$$

Given a distinguisher $\mathcal{D}$, the *total oracle query cost* of $\mathcal{D}$ is the number of queries received by the oracle E when $\mathcal{D}$ interacts with (H^E, E). Hence this is the sum of the number of direct queries of $\mathcal{D}$ to E and the number of queries made by H to E to answer $\mathcal{D}$'s queries.

Definition 2.1 (Indifferentiability [MRH04, CDMP05])**.** *A Turing machine H with oracle access to an ideal primitive E is said to be $(q, t, \sigma, \varepsilon)$ indifferentiable from an ideal primitive $\mathcal{G}$ if there exists a simulator S making at most σ many queries to*

the oracle $\mathcal{G}$ and running time at most t, such that for any adversary $\mathcal{D}$, it holds that

$$\mathsf{Adv}_D((H^E, E), (\mathcal{G}, S^{\mathcal{G}})) < \varepsilon.$$

*The total query cost of the distinguisher is at most q. H^E is said to be (*computationally*) indifferentiable from $\mathcal{G}$ if running time of $\mathcal{D}$ as well as S is bounded above by some polynomial in the security parameter k and ε is a negligible function of k.*

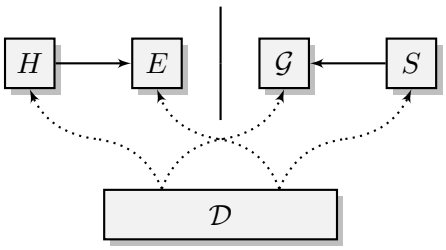

Figure 2.1: **The Indifferentiability notion:** - The distinguisher $\mathcal{D}$ either interacts with algorithm H and ideal primitive E, or with ideal primitive $\mathcal{G}$ and simulator S. Algorithm H has oracle access to E, while simulator S has oracle access to $\mathcal{G}$.

As illustrated in Figure 2.1, the role of the simulator is to simulate the ideal primitive E, so that no distinguisher can tell whether it is interacting with H and E or $\mathcal{G}$ and S. The simulator output should look "consistent" with the $\mathcal{G}$ output. Note, the simulator can not see the distinguisher's query to $\mathcal{G}$; however it can query $\mathcal{G}$ when needed. The following example illustrates the notion of indifferentiability in more details.

Example 2.3 H^E be a hash function based on a block cipher E, which is modeled as a randomly chosen block cipher or ideal cipher. The ideal primitive $\mathcal{G}$ represents a random oracle that the hash function H^E should emulate. Therefore, one obtains the following setting: the distinguisher has oracle access to both the block-cipher and the hash function. These oracles are implemented in one of the following two ways: either the block cipher E is chosen at random and the hash function is constructed from it, or the hash function is chosen at random and the block-cipher is implemented by a simulator S with oracle access to H. If H^E is indifferentiable from a random oracle, these two cases are in fact indistinguishable.

Maurer et. al. [MRH04] showed, if H^E is indifferentiable from $\mathcal{G}$, then $C^{\mathcal{G}}$ can securely replace $\mathcal{F}$ in any cryptosystem.[2] The resulting cryptosystem is at least as secure in the E model as in $\mathcal{G}$ model. For example, if a block cipher based hash function is indifferentiable from a random oracle in the ideal cipher model, then the hash function can replace the random oracle in any cryptosystem. The resulting

[2]Ristenpart et. al. [RSS11] showed, indifferentiability framework only covers cryptosystems which are secure against single entity stateful adversary. However, in this chapter by "any" cryptosystem we actually mean any cryptosystem which is secure against single entity stateful adversary.

cryptosystem remains secure in the ideal cipher model if the original scheme was secure in the random oracle model.

Following [CDMP05], we formally describe what it means for a cryptosystem to be at least as secure in the E model as in the $\mathcal{G}$ model. A cryptosystem is modeled as an Interactive Turing Machine with an interface to an adversary $\mathcal{A}$ and to a public oracle. The cryptosystem is run by an environment $\mathcal{E}$. The environment $\mathcal{E}$ also runs adversary $\mathcal{A}$ and provides a binary output in the end. In the E model (left), $\mathcal{P}$ has oracle access to H whereas $\mathcal{A}$ has oracle access to E. In the $\mathcal{G}$ model (right), both $\mathcal{P}$ and $\mathcal{A}'$ has oracle access to $\mathcal{G}$. The definition is illustrated in Figure 2.2.

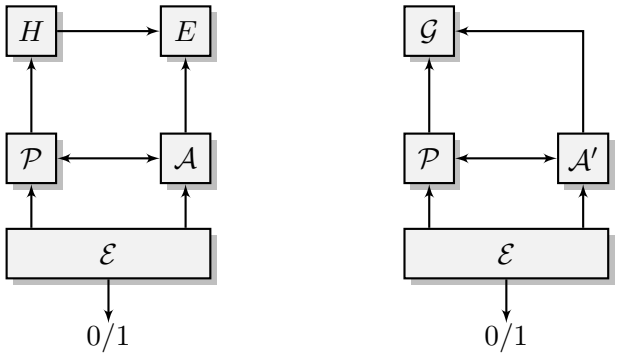

Figure 2.2: **Crypto System Models** - The environment $\mathcal{E}$ interacts with cryptosystem $\mathcal{P}$ and attacker $\mathcal{A}$. In the E model (left), $\mathcal{P}$ has oracle access to H whereas $\mathcal{A}$ has oracle access to E. In the $\mathcal{G}$ model (right), both $\mathcal{P}$ and $\mathcal{A}'$ has oracle access to $\mathcal{G}$.

Definition 2.2. *A cryptosystem $\mathcal{P}$ is said to be at least as secure in the E model with algorithm H as in the $\mathcal{G}$ model, if for any environment $\mathcal{E}$ and any attacker $\mathcal{A}$ in the E model, there exists an attacker $\mathcal{A}'$ in the $\mathcal{G}$ model, such that*

$$|\Pr[\mathcal{E}(\mathcal{P}^H, \mathcal{A}^E) \to 1] - \Pr[\mathcal{E}(\mathcal{P}^{\mathcal{G}}, \mathcal{A}'^{\mathcal{G}}) \to 1]|$$

is a negligible function of the security parameter k. Similarly, the models preserve the computational security when $\mathcal{E}$, $\mathcal{A}$ and $\mathcal{A}'$ are polynomial time in k.

The following theorem from [MRH04, CDMP05] shows that security is preserved when an ideal primitive is replaced by an indifferentiable one.

Theorem 2.1. *$\mathcal{P}$ be a cryptosystem with oracle access to an ideal primitive $\mathcal{G}$. Also, H be an algorithm such that H^E is indifferentiable from $\mathcal{G}$. Then cryptosystem $\mathcal{P}$ is at least as secure in the E model with algorithm H as in the $\mathcal{G}$ model.*

Proof. $\mathcal{E}$ be an environment interacting with the cryptosystem $\mathcal{P}$. $\mathcal{A}$ be any attacker in the E model, where $\mathcal{P}$ has oracle access to H and the attacker $\mathcal{A}$ has oracle access

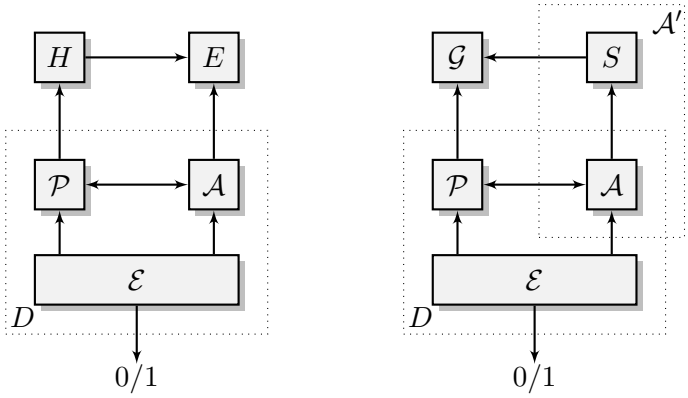

Figure 2.3: **Crypto System and Indifferentiability** - Construction of attacker $\mathcal{A}'$ from attacker $\mathcal{A}$ and simulator S.

to E. The environment $\mathcal{E}$ interacts with both $\mathcal{P}$ and $\mathcal{A}$. This is illustrated in left part of Figure 2.3.

Since H^E is indifferentiable from $\mathcal{G}$ (Figure 2.1), one can replace (H, E) by $(\mathcal{G}, S)$ with only a negligible modification of environment's output distribution.[3] As illustrated in Figure 2.3, by merging attacker $\mathcal{A}$ and simulator S one obtains an attacker $\mathcal{A}'$ in the $\mathcal{G}$ model. □

Remark 2.1 There is a minute difference between the indifferentiability definitions in Coron et. al. [CDMP05] and Maurer et. al. [MRH04]. The Definition 3.2 is actually taken from [CDMP05], where there is an universal simulator which works against any adversary (distinguisher). Where as in Maurer et. al. framework, a construction is indifferentiable as long as for any adversary there exists some simulator which can fool the adversary. That way in Maurer et. al's model, we might have different simulators which work against different adversaries. In this work, when we show a construction is indifferentiable, we always consider Definition 3.2; this implies the construction is also indifferentiable in Maurer et. al's framework. However, while showing a construction is *not* indifferentiable we consider Maurer et. al's definition; this implies the construction is also not indifferentiable in Coron et. al's framework as well.

2.3 Other Indifferentiability models

In this section we introduce a few variants of the indifferentiability model; in particular indifferentiability in *honest-but-curious* model by Dodis and Puniya [DP06] and public indifferentiability by Dodis et. al. [DRS09].

[3]The environment $\mathcal{E}$, along with the cryptosystem $\mathcal{P}$ and attacker $\mathcal{A}$ acts like an indifferentiability distinguisher D.

2.3.1 Honest but curious

The indifferentiability in the honest-but-curious model is a variant of general indifferentiability, which is considerably stronger than the classical notion of indistinguishability. For a special type of constructions called *transparent constructions*, this notion is equivalent to general indifferentiability. In this notion of indifferentiability, the distinguisher effectively has access to only one oracle. In the E model, the distinguisher can only query the construction H^E, not the primitive E. In addition, it also has access to the queries made by the construction H to E, which we will denote as the communication transcript $\mathcal{T}_{H\leftrightarrow E}$. Thus the role of the simulator S in the $\mathcal{G}$ model changes from trying to simulate E in the general indifferentiability (Definition 3.2) to trying to simulate the communication transcript $\mathcal{T}_{H\leftrightarrow E}$ in the $\mathcal{G}$ model. In the E model, the distinguisher's queries can be divided into two types. Those for which it does not observe the queries of H to E and for which it does. In the $\mathcal{G}$ model, the former queries (Type-I) are sent directly to the $\mathcal{G}$ oracle and the responses of $\mathcal{G}$ are sent back. While the latter queries (Type-II) are made through the simulator S, which forwards the same query to the $\mathcal{G}$ oracle. However, in addition to sending $\mathcal{G}$'s response to the distinguisher, it also sends a simulated communication transcript $\mathcal{T}_S$. This is explained in Figure 2.4. The following definition is taken from [DP06].

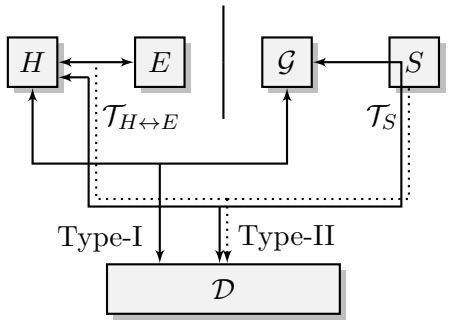

Figure 2.4: **Indifferentiability in honest-but-curious model** - The distinguisher D either interacts with H and gets the transcript $\mathcal{T}_{H\leftrightarrow E}$ or it interacts with $\mathcal{G}$ and gets the simulated transcript $\mathcal{T}_S$.

Definition 2.3. *A Turing machine H with oracle access to an ideal primitive E is said to be (q, t, ε) indifferentiable from an ideal primitive $\mathcal{G}$ in the honest-but-curious model if there exists a simulator S such that for any distinguisher D it holds that*

$$\left|\Pr[D^{H,\mathcal{T}_{H\leftrightarrow E}} = 1] - \Pr[D^{\mathcal{G},\mathcal{T}_S} = 1]\right| < \varepsilon.$$

The simulator S simulates the transcript $\mathcal{T}_S$ for Type-II queries made by the distinguisher to it and runs in at most time t . The total query cost of the distinguisher is at most q. H^E is said to be computationally indifferentiable in the honest-but-curious model from $\mathcal{G}$ if running time of D as well as S is bounded above by some polynomial in the security parameter k and ε is a negligible function of k.

Remark 2.2 Over here, the simulator S can not make any extra $\mathcal{G}$ queries, apart from forwarding the queries made by the distinguisher D.

2.3.2 Public indifferentiability

In many applications, hash functions are applied only to public messages. Such public-use occurs in many signature schemes such as, Full Domain Hash (FDH) [BR93], Probabilistic FDH [Cor02], Fiat-Shamir [FS86], Probabilistic Signature Scheme (PSS) [BR96], BLS [BLS01, BLS04]. Even some encryption schemes such as a variant of Boneh-Franklin IBE [CHK03, CHK07] and Boneh-Boyen IBE [BB04] restrict the hash function usage only to public messages. Yoneyama et al. [YMO09] and Dodis et al. [DRS09] independently realized that for these kind of cryptosystems, the random oracle model is much stronger than needed. Yoneyama et al. and Dodis et al. both proposed a weaker random oracle model which is sufficient to guarantee security of such cryptosystems; they called the weaker model as *leaky random oracle* and *public-use random oracle* respectively. Dodis et. al. introduced the notion of indifferentiability for public-use ideal primitives or *public indifferentiability* (*pub-indifferentiability* in short). In case of pub-indifferentiability, the simulator is more powerful compared to ordinary indifferentiability. For an public ideal primitive $\mathcal{G}$, the simulator S has access to the queries made by the distinguisher $\mathcal{D}$ to the primitive $\mathcal{G}$.

Formally, given an ideal primitive $\mathcal{G}$, we define the augmented ideal primitive $\overline{\mathcal{G}}$ as the primitive exposing two interfaces: the first (regular) one is same as $\mathcal{G}$, and the second is an interface `Reveal` that, when queried, returns the ordered sequence of all (regular) queries and corresponding answers made so far by any party to the regular interface. The second interface can only be used by the simulator, not by the distinguisher.

Definition 2.4. *A Turing machine H with oracle access to an ideal primitive E is said to be $(q, t, \sigma, \varepsilon)$ pub-indifferentiable from an ideal primitive $\mathcal{G}$ if there exists a simulator S, making at most σ oracle queries to the augmented ideal primitive $\overline{\mathcal{G}}$ and running time at most t, such that for any adversary $\mathcal{D}$, it holds that*

$$\mathsf{Adv}_D((H^E, E), (\mathcal{G}, S^{\overline{\mathcal{G}}})) < \varepsilon.$$

The total query cost of the distinguisher is at most q. H^E is said to be (computationally) pub-indifferentiable from $\mathcal{G}$ if running time of $\mathcal{D}$ as well as S is bounded above by some polynomial in the security parameter k and ε is a negligible function of k.

The composition theorem from [MRH04] (Theorem 2.1) still holds with public indifferentiability for cryptosystems where all messages queried by the cryptosystems to $\mathcal{G}$ can be inferred from the adversary's query during the security experiment. All the cryptosystems mentioned in the beginning of this section falls into this category.

2.4 Relation among the indifferentiability models

Public Indifferentiability is actually a weaker notion compared to general indifferentiability. In other words if H is indifferentiable from $\mathcal{G}$, that would imply H is also *publicly* indifferentiable from $\mathcal{G}$. Where as the reverse is not necessarily true. In fact, Dodis et. al. [DRS09] in their Eurocrypt '09 paper showed that the plain Merkle-Damgard mode of operation is publicly indifferentiable from a random oracle, a result which does not hold for general indifferentiability [CDMP05]. This implies that the public indifferentiability is actually a strictly weaker notion compared to the general indifferentiability.

However, the relation between *honest-but-curious* model indifferentiability and the general indifferentiability is not so simple. In the honest-but-curious model the power of the adversary as well as the simulator are restrictive compared to the general indifferentiability model. As a result, we have constructions which are indifferentiable in honest-but-curious model, but not in the general model and vice versa. Holenstein et. al. [HKT11] showed the 14-round feistel construction with random round functions is indifferentiable from a invertible random permutation. Where as, Coron et. al. [CPS08] have previously shown the feistel construction with a constant number of rounds is not sufficient for honest-but-curious model indifferentiability. In Chapter 3 we will provide a separation result from the other direction. The 2-round feistel construction as an ideal cipher domain extender is actually indifferentiable from a bigger (double domain size) ideal cipher in the honest-but-curious model. Moreover, by providing a concrete attack (Section 3.3) we will show that for the general indifferentiability only 2-rounds are not sufficient. The attack shown in Section 3.3 also hold in public indifferentiability model. Hence, honest-but-curious model indifferentiability does not necessarily imply public indifferentiability as well. Figure 2.5 illustrates the relationship among the three notions of indifferentiability.

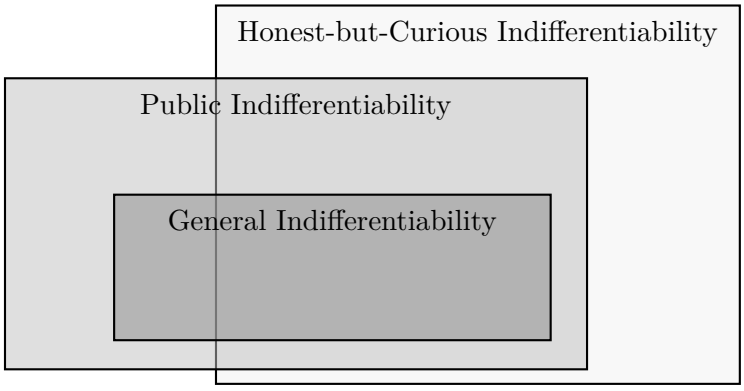

Figure 2.5: **Relation among different indifferentiability models** - General Indifferentiability implies Public Indifferentiability. However, honest-but-curious model indifferentiability do not necessarily imply general/public indifferentiability and vice versa

Chapter 3

A domain extender for the ideal cipher

We describe the first domain extender for ideal ciphers, *i.e.* we show a construction that is indifferentiable from a $2n$-bit ideal cipher, given a n-bit ideal cipher. Our construction is based on a 3-round Feistel, and is more efficient than first building a n-bit random oracle from a n-bit ideal cipher (as in [CDMP05]) and then a $2n$-bit ideal cipher from a n-bit random oracle (as in [HKT11], using a 14-round Feistel). We also show that 2 rounds are not enough for indifferentiability by exhibiting a simple attack. We also consider our construction in the standard model: we show that 2 rounds are enough to get a $2n$-bit tweakable block-cipher from a n-bit tweakable block-cipher and we show that with 3 rounds we can get beyond the birthday security bound. This is a joint work with Jean-Sébastien Coron, Yevgeniy Dodis and Yannick Seurin [CDMS10].

Contents

3.1 Introduction

A block cipher is a primitive that encrypts a n-bit string using a k-bit key. The standard security notion for block-ciphers is to be indistinguishable from a random permutation, for a polynomially bounded adversary, when the key is generated at random in $\{0,1\}^k$. A block-cipher is said to be a strong pseudo-random permutation (or chosen-ciphertext secure) when computational indistinguishability holds even when the adversary has access to the inverse permutation.

When dealing with block-ciphers, it is sometimes useful to work in an idealized model of computation, in which a concrete block-cipher is replaced by a publicly accessible random block-cipher (or ideal cipher); this is a block cipher with a k-bit key and a n-bit input/output, that is chosen uniformly at random among all block ciphers of this form; this is equivalent to having a family of 2^k independent random permutations. All parties including the adversary can make both encryption and decryption queries to the ideal block cipher, for any given key; this is called the Ideal Cipher Model (ICM). Many schemes have been proven secure in the ICM [BRS02, Des00, EM91, EM97, Gra02, Jon02, KR01, PP03]; however, it is possible to construct artificial schemes that are secure in the ICM but insecure for any concrete block cipher (see [Bla06]). Still, a proof in the ideal cipher model seems useful because it shows that a scheme is secure against generic attacks, that do not exploit specific weaknesses of the underlying block cipher.

It was shown in [CDMP05, HKT11] that the Ideal Cipher Model(ICM) and the Random Oracle Model are equivalent; the random oracle model is similar to the ICM in that a concrete hash function is replaced by a publicly accessible random function (the random oracle). The authors of [CDMP05] proved that a random oracle (taking arbitrary long inputs) can be replaced by a block cipher-based construction, and the resulting scheme will remain secure in the ideal cipher model. Conversely, it was shown in [HKT11] that an ideal cipher can be replaced by a 14-round Feistel construction, and the resulting scheme will remain secure in the random oracle model. Both directions were obtained using an extension of the classical notion of indistinguishability, called *indifferentiability*, introduced by Maurer *et al.* in [MRH04].

Since a block cipher can only encrypt a string of fixed length, one must consider the encryption of longer strings. A *mode of operation* of a block-cipher is a method used to extend the domain of applicability from fixed length strings to variable length strings. Many modes of operations have been defined that provide both privacy and authenticity (such as OCB [RBB03]). A mode of operation can also be a permutation; in this case, one obtains an extended block cipher that must satisfy the same property as the underlying block-cipher, *i.e.* it must be a (strong) pseudo-random permutation. Many constructions of domain extender for block-ciphers have been defined that satisfy this security notion, for example PEP [CS06b], XCB [MF07], HCTR [WFW05], HCH [CS06a, CS08] and TET [Hal07].

However, it is easy to see that none of those constructions provide the indifferentiability property that enables to get a $2n$-bit ideal cipher from a n-bit ideal cipher. This is because these constructions were proposed with privacy concerns in

mind (mainly for disk encryption purposes) and proven secure only in the classical pseudo-random permutation model. Therefore, these constructions cannot be used when security must hold under the random permutation model (or ideal cipher model). Consider for example the public-key encryption scheme described by Phan and Pointcheval in [PP03]. The scheme requires a public random permutation with the same size as the RSA modulus, say 1024 bits. In order to replace a 1024-bit random permutation by a construction based on a smaller primitive (for example a 128-bit block cipher), indifferentiability with respect to a 1024-bit random permutation is required. Given a 128-bit block-cipher, none of the previous constructions can provide such property; therefore if one of these constructions is plugged into the Phan and Pointcheval scheme, nothing can be said about the security of the resulting scheme.

In this work we construct the first domain extender for the ideal cipher; that is we provide a construction of an ideal cipher with $2n$-bit input from an ideal cipher with n-bit input. Given an ideal cipher with n-bit input/output, one could in principle use the construction in [CDMP05] to get a random oracle with n-bit output, and then use the 14-round Feistel in [HKT11] to obtain an ideal cipher with $2n$-bit input/output, but that would be too inefficient. Moreover the security bound in [HKT11] is rather loose, which implies that the construction only works for large values of n.[1] Over here, we describe a more efficient construction, based on a 3-round Feistel only, and with a better security bound. More precisely, we show that the 3-round construction in Figure 3.1 (left) is enough to get a $2n$-bit random permutation from a n-bit ideal cipher, and that its variant in Figure 3.1 (right) provides a $2n$-bit ideal cipher. We also show that 2 rounds are not enough by providing a simple attack. Interestingly, in the so called honest-but-curious model of indifferentiability [DP06], we show that 2 rounds are sufficient.

Our construction is similar to that of Luby-Rackoff [LR88]. However we stress that the "indifferentiable construction" security notion is very different from the classical indistinguishability notion. The well known Luby-Rackoff result that 4 rounds are enough to obtain a strong pseudo-random permutation from pseudo-random functions [LR88], is proven under the classical indistinguishability notion. Under this notion, the adversary has only access to the input/output of the Luby-Rackoff construction, and tries to distinguish it from a random permutation; in particular it does not have access to the input/output of the inner pseudo-random functions. On the contrary, in our setting, the distinguisher can make oracle calls to the inner block-ciphers E_i's (see Fig. 3.1); the indifferentiability notion enables to accommodate these additional oracle calls in a coherent definition.

The indifferentiability security notion still requires a (small) ideal component. We stress that it is unknown how to instantiate such ideal component (be it a random oracle or an ideal cipher, as opposed to a PRF or a PRP) and that the security guarantee does not hold anymore once that component is instantiated. Moreover the recent related-key attacks on AES [BKN09, BK09] show that AES-

[1] The security bound in [HKT11] for the 14-round Feistel random oracle based construction is $q^{16}/2^n$, where q is the number of distinguisher's queries. This implies that for $q = 2^{64}$, one must take at least $n = 1024$, which corresponds to a 2048-bit permutation.

192 and AES-256 do not behave as ideal ciphers; as of 2009 it is unclear if we have a candidate block-cipher with key-size larger than block-size that behaves like an ideal cipher.

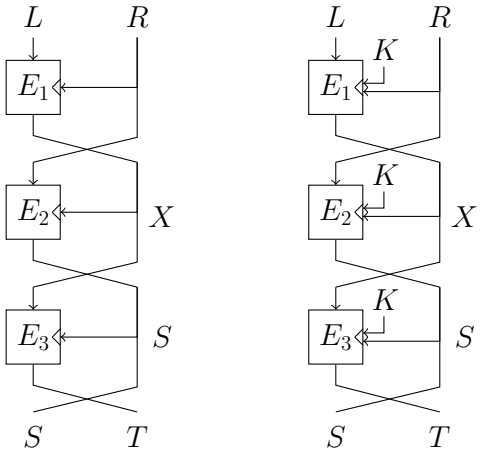

Figure 3.1: Construction of a $2n$-bit permutation given a n-bit ideal cipher with n-bit key (left). Construction of a $2n$-bit ideal cipher with k-bit key, given a n-bit ideal cipher with $(n + k)$-bit key (right).

Finally, we also analyze our construction in the standard model. In this case, we use a *tweakable* block-cipher as the underlying primitive. Tweakable block-ciphers were introduced by Liskov, Rivest and Wagner in [LRW02, LRW11] and provide an additional input - the tweak - that enables to get a *family* of independent block-ciphers; efficient constructions of tweakable block-ciphers were described in [LRW02, LRW11], given ordinary block-ciphers. Over here, we show that our construction with 2 rounds enables to get a $2n$-bit tweakable block-cipher from a n-bit tweakable block-cipher. Moreover we show that with 3 rounds we achieve a security guarantee beyond the birthday paradox.

3.1.1 Related Work

At FSE 2009, Minematsu [Min09] provided two constructions of a $2n$-bit block-cipher from an n-bit tweakable block-cipher :

1. A 3-round Feistel construction with universal hashing in the 1st round and tweakable block ciphers in the 2nd and the 3rd rounds. This construction is a secure pseudo-random permutation beyond the birthday bound.

2. A 4-round Feistel with universal hashing in the 1st and the 4th rounds and tweakable block ciphers in the 2nd and the 3rd rounds. This construction is a secure strong pseudo-random permutation beyond the birthday bound.

On the other hand, our construction in this section is a 3-round Feistel, with tweakable block ciphers in every round, and it gives a secure (tweakable) strong

pseudo-random permutation beyond the birthday bound. Therefore, the construction in [Min09] is more efficient as only 2 calls are required to the underlying tweakable block-cipher, instead of 3 calls in our construction (this is assuming very fast universal hashing, *e.g.* [Kro06]). However, we stress that the constructions in [Min09] are secure only in the symmetric-key setting; it is easy to see that none of the two constructions from [Min09] can achieve the indifferentiability property (the attack is similar to the attack against 2-round Feistel described in Section 3.3).

3.2 Basics

In this section we briefly recall a few notions regarding ideal primitives and indifferentiability from Chapter 2 (Section 2.2), which would be useful in this chapter. We first recall the notion of indifferentiability of random systems, introduced by Maurer *et al.* in [MRH04]. This is an extension of the classical notion of indistinguishability, where one or more oracles are publicly available, such as random oracles or ideal ciphers.

Ideal primitive is an algorithmic entity which receives inputs from one of the parties and delivers its output immediately to the querying party. In this chapter, we consider ideal primitives such as random oracle, random permutation and ideal cipher. A *random oracle* [BR93] is an ideal primitive which provides a random output for each new query; identical input queries are given the same answer. A *random permutation* is an ideal primitive that provides oracle access to a random permutation $P : \{0,1\}^n \rightarrow \{0,1\}^n$ and to P^{-1}. An *ideal cipher* is a generalization of a random permutation that models a random block cipher $E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$. Each key $k \in \{0,1\}^k$ defines an independent random permutation $E_k = E(k,\cdot)$ on $\{0,1\}^n$. The ideal primitive also provides oracle access to E and E^{-1}; that is, on query $(0,k,m)$, the primitive answers $c = E_k(m)$, and on query $(1,k,c)$, the primitive answers m such that $c = E_k(m)$. We stress that in the ideal cipher model, the adversary has oracle access to a publicly available ideal cipher and must send both the key and the plaintext in order to obtain the ciphertext; this is different from the standard model in which the key is privately generated by the system.

The notion of indifferentiability of random systems, introduced by Maurer *et al.* in [MRH04], is an extension of the classical notion of indistinguishability, where one or more oracles are publicly available, such as random oracles or ideal ciphers. Indifferentiability enables us to show that an ideal primitive $\mathcal{P}$ (for example, a random permutation) can be replaced by a construction C that is based on some other ideal primitive E; for example, C can be the Feistel construction illustrated in Fig. 3.1 (left). The formal definition of indifferentiability was given in Chapter 2 (Definition).

The indifferentiability notion is illustrated in Figure 3.2, where C is our 3-round construction of Figure 3.1 (left), E is an ideal cipher, $\mathcal{P}$ is a random permutation and $\mathcal{S}$ is the simulator. In this chapter, for a 3-round construction, we denote these ideal ciphers by E_1, E_2, E_3 (see Fig. 3.1). Equivalently, one can consider a single ideal cipher E and encode in the first 2 key bits which round ideal cipher E_1, E_2, or E_3 is actually called. The distinguisher has either access to the system formed by

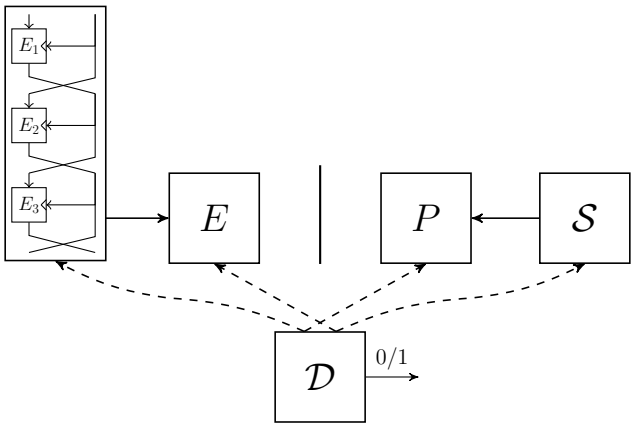

Figure 3.2: The indifferentiability notion and 3-round construction

the construction C and the ideal cipher E, or to the system formed by the random permutation P and a simulator $\mathcal{S}$. In the first system (left), the construction C computes its output by making calls to the ideal cipher E (equivalently the 3 ideal ciphers E_1, E_2 and E_3); the distinguisher can also make calls to E directly. In the second system (right), the distinguisher can either query the random permutation P, or the simulator that can make queries to P. If the distinguisher first makes a call to the construction C, and then makes the corresponding calls to ideal cipher E, he will get the same answer. This must remain true when the distinguisher interacts with permutation P and simulator $\mathcal{S}$. The role of simulator $\mathcal{S}$ is then to simulate the ideal ciphers E_i's so that 1) the output of $\mathcal{S}$ should be indistinguishable from that of ideal ciphers E_i's and 2) the output of $\mathcal{S}$ should look "consistent" with what the distinguisher can obtain independently from P. We note that in this model the simulator does not see the distinguisher's queries to P; however, it can call P directly when needed for the simulation.

The indifferentiability notion is the "right" notion for substituting one ideal primitive with a construction based on another ideal primitive. That is, if C^E is indifferentiable from an ideal primitive $\mathcal{P}$, then C^E can replace $\mathcal{P}$ in any cryptosystem, and the resulting cryptosystem is at least as secure in the E model as in the $\mathcal{P}$ model (Chapter 2, Theorem 2.1).

3.3 An Attack against 2 Rounds

In this section we show that 2 rounds are not enough when the inner ideal ciphers are publicly accessible, that is we exhibit a property for 2 rounds that does not exist for a random permutation.

Formally, the 2 round construction is defined as follows (see Fig. 3.3). Let $E_1 : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher, where $c = E_1(K, m)$ is the n-bit ciphertext corresponding to n-bit key K and n-bit input message m; let E_2 be

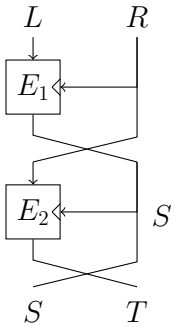

Figure 3.3: The 2-round Feistel construction $\Psi_2(L, R)$.

defined similarly. We define the permutation $\Psi_2 : \{0,1\}^{2n} \to \{0,1\}^{2n}$ as:

$$\Psi_2(L, R) := (E_1(R, L), E_2(E_1(R, L), R))$$

It is easy to see that this defines an invertible permutation over $\{0,1\}^{2n}$. Namely, given a ciphertext (S, T) the value R is recovered by "decrypting" T with block-cipher E_2 and key S, and the value L is recovered by "decrypting" S with block-cipher E_1 and key R.

The attack against permutation Ψ_2 is straightforward; it is based on the fact that the attacker can arbitrarily choose both R and S. More precisely, the attacker selects $R = 0^n$ and $S = 0^n$ and queries $L = E_1^{-1}(R, S)$ and $T = E_2(S, R)$. This gives $\Psi_2(L, R) = (S, T)$ as required. However, it is easy to see that with a random permutation P and a polynomially bounded number of queries, it is impossible to find L, R, S, T such that $P(L\|R) = S\|T$ with both $R = 0^n$ and $S = 0^n$, except with negligible probability. Therefore, the 2-round construction cannot replace a random permutation.

Theorem 3.1. *The 2-round Feistel construction Ψ_2 is not indifferentiable from a random permutation.*

In Section 3.4 we also analyze existing constructions of domain extender for block ciphers and show that they are not indifferentiable from an ideal cipher; more precisely, we show that the CMC [HR03] and EME [HR04a] constructions are not indifferentiable from an ideal cipher. We stress that our observations do not imply anything concerning their security in the standard pseudo-random permutation model.

3.4 Previous Constructions are not Indifferentiable

We analyze previous constructions of domain extender for block ciphers and show that they are not indifferentiable from an ideal cipher. This is not surprising as all these constructions were proposed with privacy concerns in mind (mainly for disk encryption purposes) and proven secure in the classical Luby-Rackoff model.

Most of this constructions use two layers of keyed universal hashing and cannot be analyzed in the indifferentiability framework: this is the case for example of PEP [CS06b], XCB [MF07], HCTR [WFW05], HCH [CS06a, CS08] and TET [Hal07].

Other constructions however use nothing more than the underlying block cipher. The two most prominent of them are CMC [HR03] and EME [HR04a] proposed by Halevi and Rogaway. We now show that these two constructions are not indifferentiable from an ideal cipher.

3.4.1 The CMC construction

CMC was proposed by Halevi and Rogaway [HR03] and uses two layers of CBC and an intermediate mixing layer. This is a tweakable mode but we don't consider the tweak in our description (that is we set the tweak to $T = 0^n$) since it is not relevant for our attack.

CMC uses a block cipher $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ and turns it into a tweakable block cipher $\mathbf{E}$ with tweak space $\{0,1\}^n$, key space $\{0,1\}^k \times \{0,1\}^k$, and message space $\bigcup_{m\geq 2}\{0,1\}^{mn}$. A message $P_1 \cdots P_m$ of m n-bit blocks is encrypted under key K, K' and tweak T as follows:

1. $\mathbb{T} \leftarrow E_{K'}(T)$
2. $PPP_0 \leftarrow \mathbb{T}$
3. for $i = 1$ to m do $PPP_i \leftarrow E_K(P_i \oplus PPP_{i-1})$
4. $M \leftarrow 2(PPP_1 \oplus PPP_m)$
5. for $i = 1$ to m do $CCC_i \leftarrow PPP_{m+i-1} \oplus M$
6. $CCC_0 \leftarrow 0^n$
7. for $i = 1$ to m do $C_i = E_K(CCC_i) \oplus CCC_{i-1}$
8. $C_1 \leftarrow C_1 \oplus \mathbb{T}$
9. return $C_1 \cdots C_m$

The attack on CMC proceeds as follows (we describe the attack for two blocks only, it can be easily extended to any number of blocks).

If first fixes two arbitrary keys K' and K, and computes $\mathbb{T} = E_{K'}(T)$. It then simply consists in computing $P_1 = E_K^{-1}(0^n)$. One can then verify that the encryption of $(P_1 \oplus \mathbb{T})|P_1$ is $\mathbf{E}_{K,K'}((P_1 \oplus \mathbb{T})\|P_1) = E_K(0)\|(E_K(0) \oplus \mathbb{T})$. Hence one has been able to find to values A and B such that $\mathbf{E}_{K,K'}((A \oplus \mathbb{T})\|A) = B\|(B \oplus \mathbb{T})$ for some fixed value $\mathbb{T}$, which would be possible with only negligible advantage for a random permutation.

3.4.2 The EME construction.

EME was proposed as CMC by Halevi and Rogaway [HR04a], and improves on CMC since it is parallelizable. It uses to layers of ECB and an intermediate mixing layer. As CMC it is tweakable but we will set the tweak to 0^n in our attack.

CMC uses a block cipher $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ and turns it into a tweakable block cipher $\mathbf{E}$ with tweak space $\{0,1\}^n$, key space $\{0,1\}^k$, and message space $\bigcup_{m\geq 2}\{0,1\}^{mn}$. A message $P_1 \cdots P_m$ of m n-bit blocks is encrypted under key K and tweak $T = 0^n$ as follows:

1. $L \leftarrow 2E_K(0^n)$
2. for $i = 1$ to m do $PPP_i = E_k(P_i \oplus 2^{i-1}L)$
3. $MP \leftarrow PPP_1 \oplus PPP_2 \oplus \cdots \oplus PPP_m$
4. $MC \leftarrow E_K(MP)$
5. $M \leftarrow MP \oplus MC$
6. for $i = 2$ to m do $CCC_i = PPP_i \oplus 2^{i-1}M$
7. $CCC_1 \leftarrow MC \oplus CCC_2 \oplus \cdots \oplus CCC_m$
8. for $i = 1$ to m do $C_i \leftarrow E_K(CCC_i) \oplus 2^{i-1}L$
9. return $C_1 \cdots C_m$

The attack on EME for two-blocks messages proceeds as follows:

1. choose an arbitrary key K and compute $L = 2E_K(0^n)$
2. compute the value MP corresponding to $MC = 0^n$, $MP = E_K^{-1}(0^n)$; note that consequently $M = MP \oplus MC = MP$
3. fix $P_1 = 0^n$ and compute $PPP_1 = E_K(P_1 \oplus L)$
4. compute $PPP_2 = MP \oplus PPP_1$ and deduce $P_2 = E_K^{-1}(PPP_2) \oplus 2L$
5. compute $CCC_1 = CCC_2 = PPP_2 \oplus 2MP$
6. compute $C_1 = E_K(CCC_1) \oplus L$ and $C_2 = E_K(CCC_2) \oplus 2L = E_K(CCC_1) \oplus 2L$

Hence this attack enables to find P_1, P_2, C_1, C_2 such that $\mathbf{E}_K(P_1 \| P_2) = C_1 \| C_2$, $P_1 = 0^n$ and $C_1 \oplus C_2 = L \oplus 2L$ for some fixed value L. This would be possible with only negligible advantage for a truly random permutation.

3.5 Indifferentiability of 3-round Feistel Construction

We now prove our first main result: the 3-round Feistel construction is indifferentiable from a random permutation. To get an ideal cipher, it suffices to prepend a key K to the 3 ideal ciphers E_1, E_2 and E_3; one then gets a family of independent random permutation, parametrised by K, *i.e.* an ideal cipher (see Fig. 3.1 for an illustration).

Formally, the 3 round permutation $\Psi_3 : \{0,1\}^{2n} \to \{0,1\}^{2n}$ is defined as follows, given block ciphers E_1, E_2 and E_3 with n-bit key (first variable) and n-bit input/output (second variable):

$$\begin{aligned} X &= E_1(R, L) \\ S &= E_2(X, R) \\ T &= E_3(S, X) \\ \Psi_3(L, R) &:= (S, T) \end{aligned}$$

The 3 round block cipher $\Psi'_3 : \{0,1\}^k \times \{0,1\}^{2n} \to \{0,1\}^{2n}$ is defined as follows, given block ciphers E_1, E_2 and E_3 with $(k+n)$-bit key and n-bit input/output:

$$\begin{aligned} X &= E_1(K\|R, L) \\ S &= E_2(K\|X, R) \\ T &= E_3(K\|S, X) \\ \Psi'_3(K, (L, R)) &:= (S, T) \end{aligned}$$

Theorem 3.2. *The 3-round Feistel construction Ψ_3 is $(q, t, \sigma, \varepsilon)$-indifferentiable from a random permutation, with $t = \mathcal{O}(qn)$, $\sigma = \mathcal{O}(q)$ and $\varepsilon = 5q^2/2^n$. The 3-round block-cipher construction Ψ'_3 is $(q, t, \sigma, \varepsilon)$-indifferentiable from an ideal cipher, with $t = \mathcal{O}(qn)$, $\sigma = \mathcal{O}(q)$ and $\varepsilon = 5q^2/2^n$.*

Proof. We only consider the 3-round permutation Ψ_3; the extension to block-cipher Ψ'_3 is straightforward. We must construct a simulator $\mathcal{S}$ such that the two systems formed by (Ψ_3, E) and $(P, \mathcal{S})$ are indistinguishable (see Fig. 3.2).

Our simulator maintains an history of already answered queries for E_1, E_2 and E_3. Formally, when the simulator answers X for a $E_1(R, L)$ query, it stores $(1, R, L, X)$ in history; the simulator proceeds similarly for E_2 and E_3 queries. We write that the simulator "simulates" $E_1(R, L) \leftarrow X$ when it first generates a random $X \in \{0,1\}^n \setminus \mathcal{B}$, where $\mathcal{B}$ is the set of already defined values for $E_1(R, \cdot)$, and then stores $(1, R, L, X)$ in history, meaning that $E_1(R, L) = X$; we use similar notations for E_2 and E_3. The distinguisher's queries are answered as follows by the simulator:

$E_1(R, L)$ query:
1. Simulate $E_1(R, L) \leftarrow X$
2. $(S, T) \leftarrow \mathsf{Adapt}(L, R, X)$
3. Return X

$E_1^{-1}(R, X)$ query
1. Simulate $E_1^{-1}(R, X) \leftarrow L$
2. $(S, T) \leftarrow \mathsf{Adapt}(L, R, X)$
3. Return L

$E_2(X, R)$ query:
1. Simulate $E_1^{-1}(R, X) \leftarrow L$
2. $(S, T) \leftarrow \mathsf{Adapt}(L, R, X)$
3. Return S

$\mathsf{Adapt}(L, R, X)$:
1. $S \| T \leftarrow P(L \| R)$
2. Store $E_2(X, R) = S$ in history
3. Store $E_3(S, X) = T$ in history.
4. Return (S, T).

The procedure for answering the other queries is essentially symmetric; we provide it for completeness:

$E_3^{-1}(S, T)$ query:
1. Simulate $E_3^{-1}(S, T) \leftarrow X$
2. $(L, R) \leftarrow \mathsf{Adapt}^{-1}(S, T, X)$
3. Return X

$E_3(S, X)$ query
1. Simulate $E_3(S, X) \leftarrow T$
2. $(L, R) \leftarrow \mathsf{Adapt}^{-1}(S, T, X)$
3. Return T

$E_2^{-1}(X, S)$ query:
1. Simulate $E_3(S, X) \leftarrow T$
2. $(L, R) \leftarrow \mathsf{Adapt}^{-1}(S, T, X)$
3. Return R

$\mathsf{Adapt}^{-1}(S, T, X)$:
1. $L \| R \leftarrow P^{-1}(S \| T)$
2. Store $E_2(X, R) = S$ in history.
3. Store $E_1(R, L) = X$ in history.
4. Return (L, R)

Finally, the simulator aborts if for some E_i and some key K, it has not defined a permutation for $E_i(K, \cdot)$; that is the simulator aborts if it has defined $E_i(K, X) = E_i(K, Y)$ for some $X \neq Y$ or it has defined $E_i^{-1}(K, X) = E_i^{-1}(K, Y)$ for some $X \neq Y$. This completes the description of the simulator.

As a consistency check, it is easy to see that if the distinguisher makes a single query for $P(L \| R)$ and then queries the simulator for $X \leftarrow E_1(R, L)$, $S \leftarrow E_2(X, R)$ and $T \leftarrow E_3(S, X)$, then the distinguisher obtains $S \| T = P(L \| R)$ as required.

We now proceed to prove that the systems (Ψ_3, E) and $(P, \mathcal{S})$ are indistinguishable. We consider a distinguisher $\mathcal{D}$ making at most q queries to the system (Ψ_3, E) or $(P, \mathcal{S})$ and outputting a bit γ. We define a sequence $\mathtt{Game}_0$, $\mathtt{Game}_1$, ... of modified distinguisher games. In the first game the distinguisher interacts with the system (Ψ_3, E). We incrementally modify the system so that in the last game the distinguisher interacts with the system $(P, \mathcal{S})$, where $\mathcal{S}$ is the previously defined simulator. We denote by S_i the event that in game i the distinguisher outputs $\gamma = 1$.

• $\mathtt{Game}_0$: the distinguisher interacts with Ψ_3 and the ideal ciphers E_i.

• $\mathtt{Game}_1$: we modify the way E_i queries are answered, without actually changing the value of the answer. We also maintain an history of already answered queries for E_1, E_2 and E_3. We proceed as follows:

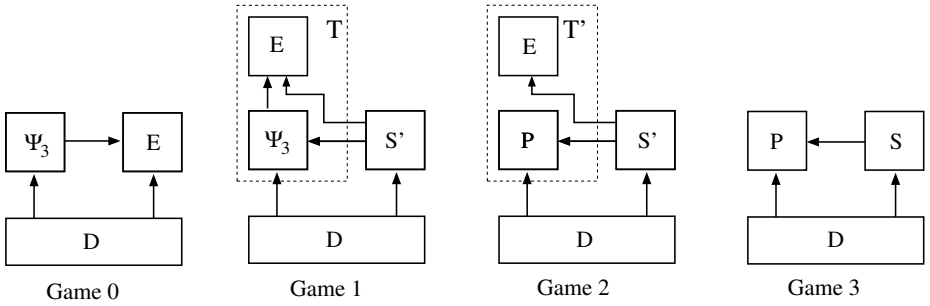

Figure 3.4: Sequence of games for proving indifferentiability.

$E_1(R, L)$ query:
1. Let $X \leftarrow E_1(R, L)$
2. $(S, T) \leftarrow \mathsf{Adapt}'(L, R, X)$
3. Return X

$E_1^{-1}(R, X)$ query
1. Let $L \leftarrow E_1^{-1}(R, X)$
2. $(S, T) \leftarrow \mathsf{Adapt}'(L, R, X)$
3. Return L

$E_2(X, R)$ query:
1. Let $L \leftarrow E_1^{-1}(R, X)$
2. $(S, T) \leftarrow \mathsf{Adapt}'(L, R, X)$
3. Return S

$\mathsf{Adapt}'(L, R, X)$:
1. $S\|T \leftarrow \Psi_3(L\|R)$
2. Store $E_2(X, R) = S$ in history.
3. Store $E_3(S, X) = T$ in history.
4. Return (S, T)

The queries to $E_2^{-1}(X, S)$, $E_3(S, X)$ and $E_3^{-1}(S, T)$ are answered symmetrically.

For example, when given a query to $E_1(R, L)$, we first query ideal cipher E_1 for $X \leftarrow E_1(R, L)$; then instead of X being returned immediately as in $\mathtt{Game}_0$, we let $S\|T = \Psi_3(L\|R)$, which gives $S = E_2(X, R)$ and $E_3(S, X) = T$; we then store $(2, X, R, S)$ and $(3, S, X, T)$ in history. Therefore, the value that get stored in history is exactly the same as the value from ideal ciphers E_2 and E_3; the only difference is that this value was obtained indirectly by querying Ψ_3 instead of directly by querying E_2 and E_3. It is easy to see that this holds for any query made by the distinguisher, who receives exactly the same answers in $\mathtt{Game}_0$ and $\mathtt{Game}_1$; this implies:

$$\Pr[S_1] = \Pr[S_0]$$

As illustrated in Fig. 3.4, we have actually constructed a simple simulator $\mathcal{S}'$ that makes queries to a subsystem $\mathcal{T}$ that comprises the construction Ψ_3 and the ideal ciphers E_1, E_2 and E_3. The difference between $\mathcal{S}'$ in $\mathtt{Game}_1$ and the main simulator $\mathcal{S}$ defined previously is that 1) $\mathcal{S}'$ calls ideal cipher $E_1(R, L)$ instead of simulating it and 2) $\mathcal{S}'$ makes calls to $\Psi_3(L\|R)$ instead of $P(L\|R)$.

• $\mathtt{Game}_2$: we modify the way the permutation queries are answered. Instead of using Ψ_3 as in system $\mathcal{T}$, we use the random permutation P in the new system $\mathcal{T}'$ (see Fig. 3.4).

We must show that the distinguisher's view has statistically close distribution in $\mathtt{Game}_1$ and $\mathtt{Game}_2$. For this, we consider the subsystem $\mathcal{T}$ with the 3-round Feistel Ψ_3 and the ideal ciphers E_i's in $\mathtt{Game}_1$, and the subsystem $\mathcal{T}'$ with the random

permutation P and ideal ciphers E_i's in $\texttt{Game}_2$. We show that the output of systems $\mathcal{T}$ and $\mathcal{T}'$ is statistically close; this in turn shows that the distinguisher's view has statistically close distribution in $\texttt{Game}_1$ and $\texttt{Game}_2$. Note that the indistinguishability of $\mathcal{T}$ and $\mathcal{T}'$ only holds for the particular set of queries made by the distinguisher and the simulator; it could not hold for any possible set of queries.

In the following, we assume that the distinguisher eventually makes a sequence of E_i queries corresponding to all previous Ψ_3 queries that he has made. More precisely, if the distinguisher has made a $\Psi_3(L, R)$ query, then eventually the distinguisher makes the sequence of queries $X \leftarrow E_1(R, L)$, $S \leftarrow E_2(X, R)$ and $T \leftarrow E_3(S, X)$ to the simulator; the same holds for $\Psi_3^{-1}(S, T)$ queries. This is without loss of generality, because from any distinguisher $\mathcal{D}$ we can build a distinguisher $\mathcal{D}'$ with the same output that satisfies this property.

The outputs to E_i queries provided by subsystem $\mathcal{T}$ in $\texttt{Game}_1$ and by subsystem $\mathcal{T}'$ in $\texttt{Game}_2$ are the same, since in both cases these queries are answered by ideal ciphers E_i. Therefore, we must show that the output to P/P^{-1} queries provided by $\mathcal{T}$ and $\mathcal{T}'$ have statistically close distribution, when the outputs to E_i queries provided by $\mathcal{T}$ or $\mathcal{T}'$ are fixed.

We consider a forward permutation query $L\|R$ made by either the distinguisher or the simulator $\mathcal{S}'$. If this $L\|R$ query is made by the distinguisher, since we have assumed that the distinguisher eventually makes the E_i queries corresponding to all his permutation queries, this $L\|R$ query will also be made by the simulator $\mathcal{S}'$, by definition of $\mathcal{S}'$. Therefore we can consider $L\|R$ queries made by the simulator $\mathcal{S}'$ only.

We first consider the answer to $S\|T = \Psi_3(L\|R)$ in $\texttt{Game}_1$. In this case the answer $S\|T$ is computed as follows:

$$\begin{aligned} X &= E_1(R, L) \\ S &= E_2(X, R) \\ T &= E_3(S, X) \end{aligned}$$

By definition of the simulator $\mathcal{S}'$, when the simulator $\mathcal{S}'$ makes a query for $\Psi_3(L\|R)$, it must have made an ideal cipher query to $E_1(R, L)$ before, or an ideal cipher query to $E_1^{-1}(R, X)$ before, with $L = E_1^{-1}(R, X)$.

If the simulator $\mathcal{S}'$ has made an ideal cipher query for $E_1(R, L)$ to subsystem $\mathcal{T}$, then from the definition of the simulator a call to $\mathsf{Adapt}'(L, R, X)$ has occurred, where $X = E_1(R, L)$; in this Adapt' call the values $E_2(X, R)$ and $E_3(S, T)$ are defined by the simulator; therefore the simulator does not make these queries to sub-system $\mathcal{T}$. This implies that the values of $E_2(X, R)$ and $E_3(S, X)$ are not included in the subsystem $\mathcal{T}$ output; therefore these values are not fixed in the probability distribution that we consider; only the value $X = E_1(R, L)$ is fixed.

Moreover, for fixed X, R the distribution of $S = E_2(X, R)$ is uniform in $\{0, 1\}^n \setminus \mathcal{B}$, where $\mathcal{B}$ is the set of already defines values for $E_2(X, \cdot)$. Since there are at most q queries, the statistical distance between the distribution of $E_2(X, R)$ and the uniform distribution in $\{0, 1\}^n$ is at most $2q/2^n$; the same holds for the distribution of $T = E_3(S, X)$. Therefore, we obtain that for a fixed X, the distribution of (S, T)

is statistically close to the uniform distribution in $\{0,1\}^{2n}$, with statistical distance at most $4q/2^n$.

If the simulator has made an ideal cipher query for $E_1^{-1}(R,X)$, then the same analysis applies and we obtain that for a fixed $L = E_1^{-1}(R,X)$ the distribution of (S,T) is statistically close to the uniform distribution in $\{0,1\}^{2n}$, with statistical distance at most $4q/2^n$. Therefore we obtain that in $\texttt{Game}_1$ the statistical distance of $S\|T = \Psi_3(L\|R)$ with the uniform distribution is always at most $4q/2^n$.

In $\texttt{Game}_2$, the output to permutation query $L\|R$ is $S\|T = P(L\|R)$; since there are at most q queries to P/P^{-1}, the statistical distance between $P(L\|R)$ and the uniform distribution in $\{0,1\}^{2n}$ is at most $2q/2^{2n}$.

Therefore the statistical distance between $\Psi_3(L,R)$ in $\texttt{Game}_1$ and $P(L\|R)$ in $\texttt{Game}_2$ is at most $4q/2^n + 2q/2^{2n} \leq 5q/2^n$. The same argument applies to inverse permutation queries. This holds for a single permutation query; since there are at most q such queries, we obtain that the statistical distance between outputs of systems $\mathcal{T}$ and $\mathcal{T}'$ to permutation queries and E_i queries, is at most $5q^2/2^n$; this implies:

$$|\Pr[S_2] - \Pr[S_1]| \leq \frac{5q^2}{2^n}$$

• $\texttt{Game}_3$: eventually the distinguisher interacts with system $(P,\mathcal{S})$. The only difference between the simulator $\mathcal{S}'$ in $\texttt{Game}_2$ and the simulator $\mathcal{S}$ in $\texttt{Game}_3$ is that instead of querying ideal ciphers E_i in $\texttt{Game}_2$, these ideal ciphers are simply simulated in $\texttt{Game}_3$, while the answer to permutation queries are exactly the same. Therefore, the distinguisher's view has the same distribution in $\texttt{Game}_2$ and $\texttt{Game}_3$, which gives:

$$\Pr[S_2] = \Pr[S_3]$$

and finally:

$$|\Pr[S_3] - \Pr[S_0]| \leq \frac{5q^2}{2^n}$$

which terminates the proof of Theorem 4.4. □

We note that the security bound in $q^2/2^n$ for our 3-round ideal cipher based construction is much better than the security bound in $q^{16}/2^n$ obtained for the 14-round Feistel construction in [HKT11] (based on random oracles).

3.5.1 Practical Considerations

Extending the Key. So far, we showed how to construct an ideal cipher Ψ_3 with $2n$-bit message and k-bit key from three ideal ciphers E_1, E_2, E_3 on n-bit message and $(n+k)$-bit key. As already mentioned, we can actually implement E_1, E_2, E_3 from a single n-bit ideal cipher E whose key length is $n+k+2$.

However, if only a block-cipher with n-bit key and n-bit message is available (for example AES-128), we need a procedure to extend the key size. To handle such cases, we notice that it suffices to first hash the key using a random oracle, and the resulting block cipher remains indifferentiable from an ideal cipher.

Lemma 3.1. *Assume* $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ *is an ideal cipher and* $H : \{0,1\}^t \to \{0,1\}^k$ *is a random oracle. Define* $E' : \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n$ *by* $E'(K', X) = E(H(K'), X)$, $E'^{-1}(K', Y) = E^{-1}(H(K'), Y)$. *Then* E' *is* $(t_D, t_S, q, \varepsilon)$*-indifferentiable from an ideal cipher, where* $t_S = \mathcal{O}(q(n+t))$ *and* $\varepsilon = \mathcal{O}(q^2/2^k)$.

Proof. We need to construct a simulator $\mathcal{S}$ for H and E, such that the two systems formed by $(E', (H, E))$ and $(\mathcal{E}', \mathcal{S})$ are indistinguishable, where $\mathcal{E}'$ is an ideal cipher with t-bit key and n-bit message.

Our simulator maintains an "H-table" of pairs (K', K) corresponding to answered queries $K = H(K')$; it also maintains an "E-table" of triples (K, X, Y) of answered queries $Y = E(K, X)$. Our simulator $\mathcal{S}$ answers the distinguisher's queries as follows:

1. $H(K')$ query: pick a random $K \leftarrow \{0,1\}^k$, record the pair (K', K) in the "H-table" and return K.

2. $E(K, X)$ query: if there exists a tuple (K, X, Y) in the E-table, return Y. Else, if there exists a tuple (K', K) in the "H-table", query the value $Y = \mathcal{E}'(K', X)$, record (K, X, Y) in the "E-table" and return Y. Else, pick a random $Y \leftarrow \{0,1\}^n$, record (K, X, Y) in the "E-table", while making sure that no collision is created for $E(K, \cdot)$; otherwise, a new Y is generated. The simulator returns Y.

3. $E^{-1}(K, Y)$ query: if there exists a tuple (K, X, Y) in the E-table (see below), return X. Else, if there exists a tuple (K', K) in the "H-table", query the value $Y = \mathcal{E}'^{-1}(K', Y)$, record (K, X, Y) in the "E-table" and return X. Else, pick a random $X \leftarrow \{0,1\}^n$, record (K, X, Y) in the "E-table", while making sure that no collision is created for $E^{-1}(K, \cdot)$, and return X.

This completes the description of the simulator. Now we show that the system $(E', (H, E))$ is indistinguishable from the system $(\mathcal{E}', \mathcal{S})$, where:

$$E'(K', X) = E(H(K'), X)$$

is the construction with extended key-size. We consider a distinguisher $\mathcal{D}$ making at most q queries and outputting a bit γ. We define a sequence $\texttt{Game}_0$, $\texttt{Game}_1$, ... of modified distinguisher games. In the first game $\texttt{Game}_0$, the distinguisher interacts with the system formed by $(\mathcal{E}', \mathcal{S})$. We denote by S_i the event in game i that the distinguisher outputs $\gamma = 1$.

$\texttt{Game}_0$: the distinguisher interacts with the simulator $\mathcal{S}$ and the ideal cipher $\mathcal{E}'$.

$\texttt{Game}_1$: we slightly modify the way H and E queries are answered by the simulator. In $\texttt{Game}_1$, given a query K' for H, instead of letting $K \leftarrow \{0,1\}^k$, the new simulator $\mathcal{S}'$ makes a query for random oracle H and returns $K = H(K')$. Similarly, for a $E(K, X)$ query, instead of generating a random $Y \leftarrow \{0,1\}^n$, the simulator queries ideal cipher E and returns $E(K, X)$; similarly for E^{-1}. Since we have simply replaced one set of random variables by a different, but identically distributed, set of random variables, we have:

$$\Pr[S_0] = \Pr[S_1]$$

$\texttt{Game}_2$: we modify the way $\mathcal{E}'$ queries are answered by the system. Instead of returning $\mathcal{E}'(K', m)$ with ideal cipher $\mathcal{E}'$, the system returns $E'(K', m) = E(H(K'), m)$ by calling ideal cipher E and random oracle H.

We must show that the distinguisher's view has statistically close distribution in $\texttt{Game}_1$ and $\texttt{Game}_2$. For this, we consider the subsystem $\mathcal{T}$ with the ideal cipher $\mathcal{E}'$ and ideal cipher E and random oracle H in $\texttt{Game}_1$, and the subsystem $\mathcal{T}'$ with construction E' and ideal cipher E and random oracle H in $\texttt{Game}_2$. We show that the output of systems $\mathcal{T}$ and $\mathcal{T}'$ is statistically close; this in turn shows that the distinguisher's view has statistically close distribution in $\texttt{Game}_1$ and $\texttt{Game}_2$.

The outputs to E queries provided by subsystem $\mathcal{T}$ in $\texttt{Game}_1$ and by subsystem $\mathcal{T}'$ in $\texttt{Game}_2$ are the same, since in both cases these queries are answered by ideal cipher E. Therefore, we must show that the output to $\mathcal{E}'$ queries provided by $\mathcal{T}$ and $\mathcal{T}'$ have statistically close distribution, when the outputs to E and H queries provided by $\mathcal{T}$ or $\mathcal{T}'$ are fixed.

We consider a $\mathcal{E}'(K', m)$ query made either by the distinguisher or by the simulator (the argument for a $\mathcal{E}'^{-1}$ query is similar). In $\texttt{Game}_2$ the answer c is computed as $E(H(K'), m)$; we have that conditioned on the event that no collision occurs for H, the output distribution of $E(H(K'), m)$ in $\texttt{Game}_2$ is exactly the same as the distribution of $\mathcal{E}'(K', m)$ in $\texttt{Game}_1$. Let denote by $\texttt{Bad}$ the event that a collision occurs for H; since there are at most q queries from the distinguisher, we have:

$$\Pr[\texttt{Bad}] \leq \frac{q^2}{2^k}$$

and we obtain:

$$|\Pr[S_2] - \Pr[S_1]| \leq \Pr[\texttt{Bad}] \leq \frac{q^2}{2^k}$$

$\texttt{Game}_3$: the distinguisher interacts with system $(E', (H, E))$. We have that the system $(E', (H, E))$ provides the same output as the system in $\texttt{Game}_2$, which gives:

$$\Pr[S_3] = \Pr[S_2]$$

From the previous inequalities, we obtain the following upper bound on the distinguisher's advantage:

$$|\Pr[S_3] - \Pr[S_0]| \leq \frac{q^2}{2^k}$$

which terminates the proof of Lemma 3.1.

□

Using this observation, given a single ideal cipher E on n-bit messages and k-bit key and a random oracle H with output size k bits, we can first build an ideal cipher E' with n-bit message and $(n + k' + 2)$-bit key, and then from Theorem 4.4 we can obtain an ideal cipher Ψ_3 on $2n$-bit messages and k'-bit key. It remains to remove the assumption of having random oracle H; this can easily be accomplished by sacrificing 1 key bit from E, and then using one of the two resulting (independent) ideal ciphers to efficiently implement H using any of the methods from [CDMP05].

GOING BEYOND DOUBLE? Another natural question is to extend the domain of the ideal cipher beyond doubling it. One way to accomplish this task is to apply our 3-round construction recursively, each time doubling the domain. However, in this case it is not hard to see that, to extend the domain by a factor of t, the original block cipher E will have to be used $\mathcal{O}(t^{\log_2 3})$ times.[2] This makes the resulting constructions somewhat impractical for large t.

In contrast, assume that we use the 2-step construction: first build a length-preserving random oracle H on $nt/2$ bits (using [CDMP05]), and then use the 14-round Feistel construction [HKT11] to get a nt-bit permutation. To construct a random oracle from $nt/2$-bit to $nt/2$-bit, only $\mathcal{O}(t)$ calls to the n-bit ideal cipher are required (first hash from $nt/2$-bit to n-bit using [CDMP05], then expand back to $nt/2$-bits using counter mode). Therefore the 2-step construction requires only $\mathcal{O}(t)$ calls to E, instead of $\mathcal{O}(t^{\log_2 3})$ when iterating our construction. This implies that for large t, the 2-step construction is more efficient.

To give a practical example, let us consider the applications of [Gra02, PP03], where one needs to apply a random permutation to the domain of an RSA modulus. We take the length of modulus N to be 1024 bits and the underlying block-cipher E to be $n = 128$ with 128-bit key (as in AES-128). One can see that to obtain a 1024-bit permutation from E, only $14 \times (1024/128) = 112$ calls to E are required for the 2-step construction, instead of 243 when iterating our construction. However for 1024-bit, the exact security of the 2-step construction is dominated by the term $\mathcal{O}(q^{16}/2^{512})$ from [HKT11], which requires $q \ll 2^{32}$, whereas the exact security of the recursive construction is $\mathcal{O}(q^2/2^{128})$, which requires $q \ll 2^{64}$. Therefore, for a 1024-bit permutation our recursive construction still provides a better security bound; however, for any size larger than 2048 bits, the two constructions have the same $q \ll 2^{64}$ bound [3].

To summarize, our construction is more efficient than the 2-step construction when doubling only once ($t = 2$). However for a large expansion factor t the 2-step construction is more efficient than the recursive method.

3.5.2 Indifferentiability for 2 Rounds in the Honest-but-curious Model

In this section we also consider the *honest-but-curious* model of indifferentiability introduced by Dodis and Puniya [DP06], which is a variant of the general indifferentiability model. We show that in the honest-but-curious model, 2 rounds as depicted in Fig 3.3 are actually sufficient to get indifferentiability.

First, we briefly recall the model; for more details we refer to [DP06]. In the honest-but-curious model of indifferentiability, the distinguisher cannot make direct

[2] In essence, this is because we call E three times for each doubling. Actually, this is not counting the calls to the independent variable length random oracle H to hash down the key, as above. However, because the constructions of such an H in [CDMP05] are so efficient, it is not hard to see that, even when implementing H using E itself, the dominant term remains $\mathcal{O}(t^{\log_2 3})$ (although the constant is slightly worse).

[3] The length-preserving random oracle used in the 14-round Feistel has the birthday bound of $q^2/2^{128}$

queries to the inner primitive E. Instead it can only query the global construction C and get the results of the internal queries made by the construction to the inner primitive E. There are actually two types of queries made by the distinguisher: those for which it asks for the transcript of the queries made by the construction to the primitive E, and those for which it does not. When the distinguisher interacts with $(\mathcal{P}, \mathcal{S}^{\mathcal{P}})$, the second queries are sent directly to $\mathcal{P}$ (and are not seen by the simulator), while the first ones are sent to the simulator $\mathcal{S}$, which must simulate the transcript of the construction's inner queries to E. Another important difference with general indifferentiability is that here the simulator cannot make its own additional queries to $\mathcal{P}$.

Theorem 3.3. *The 2-round construction is (q, t, ϵ)-indifferentiable in the honest-but-curious model from a random permutation, with $t = \mathcal{O}(qn)$ and $\epsilon = 2q^2/2^n$, where q is the total number of distinguisher queries and n is the domain size of the inner ciphers.*

Proof. We restrict ourself to distinguishers which do not make twice the same query (or the inverse query corresponding to a previous query). Note however that the distinguisher could query $L\|R$ first as a type I query (*i.e.* without asking for the transcript, and not seen by the simulator) and then as a type II query (when asking for the transcript, and sent to the simulator).

We first describe our simulator $\mathcal{S}$. It maintains an history of already defined values for E_1 and E_2. Upon a query of the distinguisher, it runs as follows:

- on input a direct query $(+, L\|R)$:
  1. query $P(L\|R) = S\|T$
  2. if $E_1(R, L)$ or $E_1^{-1}(R, S)$ is already defined, abort
  3. else $E_1(R, L) \leftarrow S$ and add $E_1(R, L) = S$ to the history
  4. if $E_2(S, R)$ or $E_2^{-1}(S, T)$ is already defined, abort
  5. else $E_2(S, R) \leftarrow T$ and add $E_2(S, R) = T$ to the history
  6. return $E_1(R, L) = S$, $E_2(S, R) = T$
- on input an inverse query $(-, S\|T)$:
  1. query $P^{-1}(S\|T) = L\|R$
  2. if $E_2(S, R)$ or $E_2^{-1}(S, T)$ is already defined, abort
  3. else $E_2^{-1}(S, T) \leftarrow R$ and add $E_2(S, R) = T$ to the history
  4. if $E_1(R, L)$ or $E_1^{-1}(R, S)$ is already defined, abort
  5. else $E_1^{-1}(R, S) \leftarrow L$ and add $E_1(R, L) = S$ to the history
  6. return $E_2^{-1}(S, T) = R$, $E_1^{-1}(R, S) = L$

We prove the indifferentiability through a sequence of games $\texttt{Game}_i$. We will note S_i the event that the distinguisher outputs 1 in $\texttt{Game}_i$. We start with:

$\texttt{Game}_0$: the distinguisher $\mathcal{D}$ interacts with $(P, \mathcal{S})$

$\texttt{Game}_1$: it is similar to $\texttt{Game}_0$ except that P now returns uniformly random answers. Looking at $\mathcal{D}$ and $\mathcal{S}$ as a distinguisher $\mathcal{D}'$ making at most q queries to P, it is easy to see that

$$|\Pr[S_1] - \Pr[S_0]| \leq \frac{q^2}{2 \cdot 2^{2n}}.$$

$\texttt{Game}_2$: we modify the way the answers to type I queries (those not seen by the simulator $\mathcal{S}$) are computed. Instead of being asked directly to the permutation P, they are "intercepted" by an algorithm $\mathcal{M}$ which forwards them to the simulator $\mathcal{S}$. $\mathcal{M}$ then computes the answer to $\mathcal{D}$ using the values returned by $\mathcal{S}$.

As long as the simulator does not abort, the output of $\mathcal{M}$ in $\texttt{Game}_2$ is the same as the output of P in $\texttt{Game}_1$. Moreover as long as the simulator does not abort, its output is also the same in $\texttt{Game}_2$ as in $\texttt{Game}_1$ since it does not depend on the additional queries made by $\mathcal{M}$. Hence:

$$|\Pr[S_2] - \Pr[S_1]| \leq \Pr_{\texttt{Game}_2}[\mathcal{S} \text{ aborts}].$$

Let $\texttt{bad}$ denote the event that there exists $1 \leq j < i \leq q$ such that the i-th and j-th queries of the distinguisher are such that

$$(R_i = R_j) \wedge (S_i = S_j) \wedge (L_i \neq L_j \vee T_i \neq T_j).$$

It is easy to see that as long as $\texttt{bad}$ does not happen, the simulator does not abort since it is always able to define the values of the internal ciphers. Therefore:

$$\Pr_{\texttt{Game}_2}[\mathcal{S} \text{ aborts}] \leq \Pr_{\texttt{Game}_2}[\texttt{bad}]$$

Moreover, defining $\texttt{bad}_i$ as the event that $\texttt{bad}$ happens exactly at the i-th query of the distinguisher, we get:

$$\Pr_{\texttt{Game}_2}[\texttt{bad}] = \sum_{i=1}^{q} \Pr_{\texttt{Game}_2}[\texttt{bad}_\texttt{i}]$$

Assume that the i-th query is a direct one: $(+, L_i|R_i)$; the argument for inverse queries is similar. Note that this query cannot have been done to P yet. Since there are at most $i-1$ values S_j in the history of P and since P returns uniformly random answers, we obtain:

$$\Pr_{\texttt{Game}_2}[\texttt{bad}_\texttt{i}] \leq \frac{i-1}{2^n}$$

which gives:

$$\Pr_{\texttt{Game}_2}[\texttt{bad}] = \sum_{i=1}^{q} \Pr_{\texttt{Game}_2}[\texttt{bad}_\texttt{i}] \leq \frac{q^2}{2 \cdot 2^n}$$

and eventually:

$$|\Pr[S_2] - \Pr[S_1]| \leq \frac{q^2}{2 \cdot 2^n}.$$

$\texttt{Game}_3$: we remove the permutation P and modify the simulator into a new simulator $\mathcal{S}'$ which, upon reception of a direct query $L\|R$, defines $S = E_1(R, L)$ uniformly at random and $T = E_2(S, R)$ uniformly at random, and symmetrically for inverse queries. Looking at $\mathcal{D}$ and $\mathcal{M}$ as a distinguisher $\mathcal{D}'$, one can see that the output of $\mathcal{S}$ in $\texttt{Game}_2$ and $\mathcal{S}'$ in $\texttt{Game}_3$ are exactly the same, which gives:

$$\Pr[S_3] = \Pr[S_2]$$

$\texttt{Game}_4$: the distinguisher interacts with the construction and the ideal ciphers E_1, E_2. We have that $\texttt{Game}_3$ and $\texttt{Game}_4$ are identical unless some collision happens in $\texttt{Game}_3$ when defining two values for the same key. Hence:

$$|\Pr[S_4] - \Pr[S_3]| \leq 2\frac{q^2}{2 \cdot 2^n} = \frac{q^2}{2^n}.$$

Putting everything together yields

$$|\Pr[S_4] - \Pr[S_0]| \leq 2\frac{q^2}{2 \cdot 2^{2n}} + \frac{q^2}{2^n} \leq \frac{2q^2}{2^n}.$$

□

Remark 3.1 Indifferentiability in the honest-but-curious model has been shown to imply indifferentiability in the general model for so-called transparent constructions [DP06]. A construction is said to be transparent if there exists an efficient algorithm which can compute the value of the inner primitive E on any input x by making a polynomial number of queries to the construction and receiving the transcript of the inner queries of the construction to E. Since the 2-round construction is not indifferentiable in the general model, this shows that it is also not transparent: namely it is impossible to efficiently compute $E_2(S, R)$ for some arbitrary value S, or $E_1^{-1}(R, S)$ for some arbitrary value R, given only oracle access + transcript to $\Psi_2(L, R)$ and $\Psi_2^{-1}(S, T)$.

3.6 Domain Extension of Tweakable Block Cipher

In this section, we also analyse our construction in the standard model, and we use a *tweakable* block-cipher as the underlying primitive. The main result of this section is that a 3-round Feistel enables to get a security guarantee beyond the birthday paradox.

Tweakable block-ciphers were introduced by Liskov, Rivest and Wagner in [LRW02, LRW11] and provide an additional input - the tweak - that enables to get a *family* of independent block-ciphers. Efficient constructions of tweakable block-ciphers were described in [LRW02, LRW11], given ordinary block-ciphers.

Definition 3.1. *A tweakable block-cipher is an efficiently computable function* $\tilde{E} : \{0,1\}^k \times \{0,1\}^\omega \times \{0,1\}^n \to \{0,1\}^n$ *that takes as input a key* $K \in \{0,1\}^k$*, a tweak* $W \in \{0,1\}^\omega$ *and a message* $m \in \{0,1\}^n$ *and returns a ciphertext* $c \in \{0,1\}^n$*. For every* $K \in \{0,1\}^k$ *and* $W \in \{0,1\}^\omega$*, the function* $\tilde{E}(K, W, \cdot)$ *is a permutation over* $\{0,1\}^n$*.*

The security notion for a tweakable block-cipher is a straightforward extension of the corresponding notion for block-ciphers. A classical block-cipher E is a strong pseudo-random permutation if no adversary can distinguish $E(K, \cdot)$ from a random permutation, where $\mathcal{A}$ can make calls to both E and E^{-1}, and $K \leftarrow \{0,1\}^k$. For tweakable block-ciphers, the adversary can additionally choose the tweak, and $E(K, \cdot, \cdot)$ should be indistinguishable from a *family* of random permutations, parametrised by $W \in \{0,1\}^\omega$:

Definition 3.2. *A tweakable block-cipher is said to be (t, q, ε)-secure if for any adversary $\mathcal{A}$ running in time at most t and making at most q queries, the adversary's advantage in distinguishing $\tilde{E}(K, \cdot, \cdot)$ with $K \leftarrow \{0,1\}^k$ from a family of independent random permutation $\tilde{\Pi}(\cdot, \cdot)$ is at most ε, where $\mathcal{A}$ can make calls to both $\tilde{E}$ and $\tilde{E}^{-1}$.*

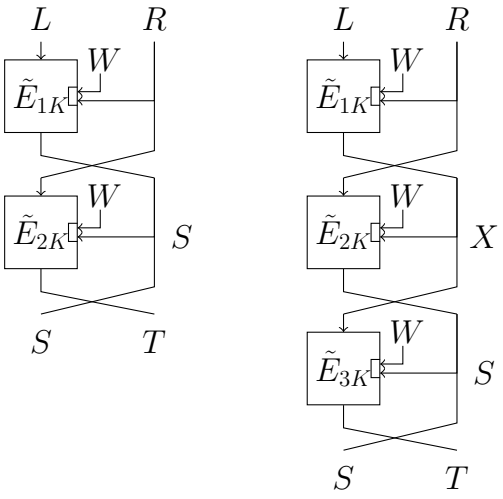

Figure 3.5: The tweakable block ciphers $\tilde{\Psi}_2$ (left) and $\tilde{\Psi}_3$ (right), with key K and tweak W

We first show that 2 rounds are enough to get a $2n$-bit tweakable block-cipher from a n-bit tweakable block-cipher (see Fig. 3.5, left). Formally, our 2-round domain extender for tweakable block-cipher works as follows. Let E_1 and E_2 be two tweakable block-ciphers with the same signature:

$$\tilde{E}_i : \{0,1\}^k \times \{0,1\}^\omega \times \{0,1\}^n \to \{0,1\}^n$$

The tweakable block cipher $\tilde{\Psi}_2 : \{0,1\}^k \times \{0,1\}^{\omega-n} \times \{0,1\}^{2n} \to \{0,1\}^{2n}$ is then defined as follows; the difference with Fig. 3.3 is that the R and S inputs go to the tweak (concatenated with the main tweak W) instead of the key.

$$\begin{aligned} S &= E_1(K, W\|R, L) \\ T &= E_2(K, W\|S, R) \\ \tilde{\Psi}_2(K, W, (L, R)) &= (S, T) \end{aligned}$$

Theorem 3.4. *The tweakable block-cipher $\tilde{\Psi}_2$ is a (t', q, ε')-secure tweakable block-cipher, if $\tilde{E}_1$ and $\tilde{E}_2$ are both (t, q, ε)-secure tweakable block-ciphers, where $\varepsilon' = 2 \cdot \varepsilon + q^2/2^n + q^2/2^{2n}$ and $t' = t - \mathcal{O}(qn)$.*

Proof. We consider an adversary making a sequence of exactly q queries. There are two types of queries $\mathcal{A}$ can make: either $(+, W, L, R)$ which is a query to $\tilde{\Psi}_2(K, W, L\|R)$, or $(-, W, S, T)$ which is a query to $\tilde{\Psi}_2^{-1}(K, W, S\|T)$. For the i-th query, we denote the by (W, L_i, R_i, S_i, T_i) the corresponding 5-uple.

$\texttt{Game}_0$: the queries are answered using $\tilde{\Psi}_2$, as illustrated in Fig. 3.5.

$\texttt{Game}_1$: we replace the tweakable block-ciphers E_1 and E_2 by 2 independent family of random permutations. From an attacker against $\tilde{\Psi}_2$ running in time t', we can construct an attacker against E_1 or E_2 running in time at most:

$$t = t' + \mathcal{O}(qn)$$

Since by assumption E_1 and E_2 are both (t, q, ε)-secure, we must have:

$$|\Pr[S_1] - \Pr[S_0]| \leq 2 \cdot \varepsilon$$

$\texttt{Game}_2$: the queries are now answered using the following process R. Given the i-th query:

1. If $(+, W, L, R)$ is queried and for some $1 \leq j < i$ the j-th 4-uple is (W, L, R, S, T), then $S\|T$ is answered.
2. If $(-, W, S, T)$ is queried and for some $1 \leq j < i$ the j-th 4-uple is (W, L, R, S, T), then $L\|R$ is answered.
3. If neither 1 nor 2 holds, then a uniformly distributed $2n$-bit string is returned.

We denote by $\texttt{Bad}$ the following event: there exists $1 \leq i < j \leq q$ such that the i-th answer $(W_i, L_i, R_i, S_i, T_i)$ and the j-th answer $(W_j, L_j, R_j, S_j, T_j)$ satisfy one of the following conditions:

1. $W_i = W_j$ and $R_i = R_j$ and $L_i \neq L_j$ and $S_i = S_j$
2. $W_i = W_j$ and $L_i = L_j$ and $R_i = R_j$ and $S_i \neq S_j$
3. $W_i = W_j$ and $S_i = S_j$ and $T_i = T_j$ and $R_i \neq R_j$

We have that conditioned on $\neg\texttt{Bad}$, the output of R in $\texttt{Game}_2$ has the same distribution as the output of $\tilde{\Psi}_2$ in $\texttt{Game}_1$, which gives:

$$\Pr[S_2|\neg\texttt{Bad}] = \Pr[S_1]$$

Moreover we have that $\Pr[\texttt{Bad}] \leq q^2/2^n$, which gives using the Difference Lemma [Sho04]:

$$|\Pr[S_2] - \Pr[S_1]| \leq \Pr[\texttt{Bad}] \leq \frac{q^2}{2^n}$$

$\texttt{Game}_3$: the adversary interacts with a family of random permutation $\tilde{\Pi}'$. We consider the following event $\texttt{Bad}'$ in $\texttt{Game}_2$: there exists $1 \leq i < j \leq q$ such that the i-th answer $(W_i, L_i, R_i, S_i, T_i)$ and the j-th answer $(W_j, L_j, R_j, S_j, T_j)$ satisfy one of the following conditions:

1. $W_i = W_j$ and $(L_i, R_i) = (L_j, R_j)$ and $(S_i, T_i) \neq (S_j, T_j)$
2. $W_i = W_j$ and $(L_i, R_i) \neq (L_j, R_j)$ and $(S_i, T_i) = (S_j, T_j)$

We have that conditioned on $\neg\mathtt{Bad}'$, the distribution of R in $\mathtt{Game}_2$ and the distribution of P in $\mathtt{Game}_3$ are the same; therefore:

$$\Pr[S_2|\neg\mathtt{Bad}'] = \Pr[S_3]$$

Moreover, we have $\Pr[\mathtt{Bad}'] \leq q^2/2^{2n}$, which gives:

$$|\Pr[S_3] - \Pr[S_2]| \leq \Pr[\mathtt{Bad}'] \leq \frac{q^2}{2^{2n}}$$

Combining the previous inequalities, we get:

$$|\Pr[S_3] - \Pr[S_0]| \leq 2 \cdot \varepsilon + \frac{q^2}{2^n} + \frac{q^2}{2^{2n}}$$

Therefore we can take $\varepsilon' = 2 \cdot \varepsilon + q^2/2^n + q^2/2^{2n}$, which terminates the proof of Theorem 3.4.

□

Now we consider the 3 round tweakable block cipher $\tilde{\Psi}_3$, defined in a similar manner as $\tilde{\Psi}_2$ (see Fig. 3.5 for an illustration). The 3-round construction enables to go beyond the birthday security bound. Namely instead of having a bound in $q^2/2^n$ as in the 2-round construction, the bound for the 3-round construction is now $q^2/2^{2n}$, which shows that the construction remains secure until $q < 2^n$ instead of $q < 2^{n/2}$.

Theorem 3.5. *The 3-round block-cipher construction Ψ_3 (see Figure 3.6) is ϵ-indistinguishable from an ideal cipher with $\epsilon = (\frac{q}{2^n})^2$ for an attacker making q block-cipher queries with $q < 2^n$.*

Proof. E_i's are actually random permutation such that $E_i : Y \times Y \to Y$. So, $\Psi_3, \tilde{\Pi}' : Y \times Y \to Y \times Y$. Here $Y = \{0,1\}^n$. The $(i+1)^{th}$ query can either be a forward permutation query or a backward permutation query. Without loss of generality we can assume if $(i+1)^{th}$ query is a forward query (L_{i+1}, R_{i+1}) is distinct from (L, R) tuples in previous queries (responses), and similarly for a backward query (S_{i+1}, T_{i+1}) is distinct from (S, T) tuples in previous queries (responses). Whether the attacker interacts with Ψ_3 or ideal cipher $\tilde{\Pi}'$, input collision means output collision and output collision means input collision. So we can also assume $(i+1)^{th}$ output pair (s_{i+1}, t_{i+1}) is distinct from previous output pairs, previous input pairs are distinct among themselves and previous output pairs are distinct among themselves.

When the attacker interacts with Ψ_3 after i queries the underlying permutations E_1, E_2, E_3 have been fixed at some points, and at other points E_1, E_2, E_3's behave randomly. Also input-output of j^{th} query is actually a 4-tuple (L_j, R_j, s_j, t_j). We

let $\mathcal{V}_i = ((L_1, R_1, s_1, t_1), \cdots, (L_i, R_i, s_i, t_i))$ be the attacker view after making the i^{th} query.

To prove Theorem 3.5 we will use the following lemma which shows for $i = 1, \cdots, q-1$ the advantage for $(i+1)^{th}$ query is actually bounded by $\frac{2i}{|Y|^2 - i}$.

Lemma 3.2. *For $i \in \{1, \cdots, q-1\}$, $\mathtt{Adv}_{i+1}$ be the distinguishing advantage for the $(i+1)^{th}$ query, then,*

$$\mathtt{Adv}_{i+1} = \frac{1}{2} \sum_{(s,t) \in Y^2 \setminus \mathcal{OP}_i} \left| \Pr[\Psi_3(L_{i+1}, R_{i+1}) = (s,t) | V_i] - \Pr[\tilde{\Pi}'(L_{i+1}, R_{i+1}) = (s,t) | V_i] \right|$$
$$\leq \frac{2i}{|Y|^2 - i}$$

where $\mathcal{OP}_i = \{(s_1, t_1), \cdots, (s_i, t_i)\}$.

If for any attacker making q queries $\mathtt{Adv}(q)$ is the distinguishing advantage, then it is not hard to show that $\mathtt{Adv}(q) \leq \sum_{i=1}^{q-1} \mathtt{Adv}_{i+1}$. Hence by Lemma 3.2 we get,

$$\mathtt{Adv}(q) \leq \sum_{i=1}^{q-1} \mathtt{Adv}_{i+1} \leq \sum_{i=1}^{q-1} \frac{2i}{|Y|^2 - i} < \sum_{i=1}^{q-1} \frac{2i}{|Y|^2 - q} = \frac{q^2 - q}{|Y|^2 - q} < \frac{q^2}{|Y|^2}$$
$$\text{As } q < |Y| = 2^n$$

□

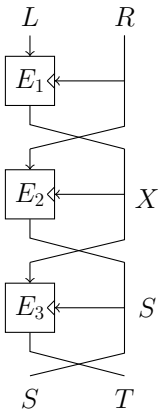

Figure 3.6: The 3-round Feistel construction $\Psi_3(L, R)$.

Proof of Lemma 3.2

We will give a proof when the $(i+1)^{th}$ query is forward permutation query, for backward permutation query the proof works in a similar fashion. From the attacker point of view

$$\bar{X} = (X_1, \cdots, X_i) = (E_1(R_1, L_1), \cdots, E_1(R_i, L_i))$$

is actually a random variable which *satisfies* $\mathcal{V}_i$.

Now we say any i-tuple $\bar{x} = (x_1, \cdots, x_i)$ is *feasible* if $\Pr[\bar{X} = \bar{x}|\mathcal{V}_i]$ is non zero. $\mathcal{F}$ be the set of all feasible $\bar{x}$. Now we will state another lemma, loosely speaking which gives an estimate of $\mathtt{Adv}_{i+1}$ for an fixed $\bar{x} \in \mathcal{F}$.

Lemma 3.3. *For all* $\bar{x} \in \mathcal{F}$,

$$\sum_{(s,t)\in Y^2\setminus\mathcal{OP}_i} \big|\Pr[\Psi_3(L_{i+1}, R_{i+1}) = (s,t)|\mathcal{V}_i \wedge \bar{X} = \bar{x}] - \Pr[\tilde{\Pi}'(L_{i+1}, R_{i+1}) = (s,t)|\mathcal{V}_i]\big| \leq \frac{4i}{|Y|^2 - i}$$

Lemma 3.3 actually almost immediately proves Lemma 3.2 as follows,

$$\begin{aligned}
&\mathtt{Adv}_{i+1} \\
&= \frac{1}{2} \sum_{(s,t)\in Y^2\setminus\mathcal{OP}_i} \big|\Pr[\Psi_3(L_{i+1}, R_{i+1}) = (s,t)|\mathcal{V}_i] - \Pr[\tilde{\Pi}'(L_{i+1}, R_{i+1}) = (s,t)|\mathcal{V}_i]\big| \\
&\leq \frac{1}{2} \sum_{(s,t)\in Y^2\setminus\mathcal{OP}_i} \sum_{\bar{x}\in\mathcal{F}} \big|\Pr[\Psi_3(L_{i+1}, R_{i+1}) = (s,t)|\mathcal{V}_i \wedge \bar{X} = \bar{x}] \\
&\qquad\qquad - \Pr[\tilde{\Pi}'(L_{i+1}, R_{i+1}) = (s,t)|\mathcal{V}_i]\big| \times \Pr[\bar{X} = \bar{x}|\mathcal{V}_i] \\
&= \frac{1}{2} \sum_{\bar{x}\in\mathcal{F}} \sum_{(s,t)\in Y^2\setminus\mathcal{OP}_i} \big|\Pr[\Psi_3(L_{i+1}, R_{i+1}) = (s,t)|\mathcal{V}_i \wedge \bar{X} = \bar{x}] \\
&\qquad\qquad - \Pr[\tilde{\Pi}'(L_{i+1}, R_{i+1}) = (s,t)|\mathcal{V}_i]\big| \times \Pr[\bar{X} = \bar{x}|\mathcal{V}_i] \\
&\leq \frac{1}{2} \sum_{\bar{x}\in\mathcal{F}} \frac{4i}{|Y|^2 - i} \times \Pr[\bar{X} = \bar{x}|\mathcal{V}_i] = \frac{2i}{|Y|^2 - i}
\end{aligned}$$

□

Proof of Lemma 3.3

At first we would like define some notations and state some important observations. For a feasible i-tuple $\bar{x} = (x_1, \cdots, x_i)$ we define

$$\mathcal{X}_{\bar{x}}^j = \{\alpha \in Y | \alpha \textit{ appears in } \bar{x} \textit{ exactly } j \textit{ times }\}.$$

Note,

$$\sum_{j=1}^{i} j|\mathcal{X}_{\bar{x}}^j| = i \tag{3.1}$$

Similarly considering the tuple $\bar{s} = (s_1, \cdots, s_i)$, we define

$$\mathcal{S}^j = \{\alpha \in Y | \alpha \textit{ appears in } \bar{s} \textit{ exactly } j \textit{ times }\}.$$

Also we have,

$$\sum_{j=1}^{i} j|\mathcal{S}^j| = i \tag{3.2}$$

When the attacker is making $(i+1)^{th}$ query the tuple $\bar{s}$ is already fixed. So we do not include the subscript $\bar{s}$ in the definition of $\mathcal{S}^j$. We define,

$$\mathcal{X}_{\bar{x}} = \bigcup_{j=1}^{i} \mathcal{X}_{\bar{x}}^j \qquad and \qquad \mathcal{S} = \bigcup_{j=1}^{i} \mathcal{S}^j$$

Hence we also have,

$$|\mathcal{X}_{\bar{x}}| = \sum_{j=1}^{i} |\mathcal{X}_{\bar{x}}^j| \qquad and \qquad |\mathcal{S}| = \sum_{j=1}^{i} |\mathcal{S}^j| \tag{3.3}$$

For the query (L_{i+1}, R_{i+1}), we say X_{i+1} is *new* with respect to $\bar{x}$ if $X_{i+1} \notin \mathcal{X}_{\bar{x}}$.We also say X_{i+1} is *k-collision* with respect to $\bar{x}$ if $X_{i+1} \in \mathcal{X}_{\bar{x}}^k$.

Now for a fixed $\bar{x} = (x_1, \cdots, x_i)$, depending on the value of X_{i+1} we define $\mathcal{S}'_{\bar{x}}(X_{i+1}) \subseteq \mathcal{S}$ as follows.

$$\mathcal{S}'_{\bar{x}}(X_{i+1}) = \{\alpha \in Y | \alpha = s_j \text{ and } x_j = X_{i+1} \text{ for some } j \in [1, i]\}$$

Intuitively $\mathcal{S}'_{\bar{x}}(X_{i+1})$ is the set of fixed outputs for $E_2(X_{i+1}, .)$. If X_{i+1} is new then $\mathcal{S}'_{\bar{x}}(X_{i+1})$ is empty, and if $X_{i+1} \in \mathcal{X}_{\bar{x}}^k$ then $|\mathcal{S}'_{\bar{x}}(X_{i+1})| = k$. This is true because if $|\mathcal{S}'_{\bar{x}}(X_{i+1})| < k$, then we would have $x_{j_1} = x_{j_2} = X_{i+1}$ and $s_{j_1} = s_{j_2}$ for some $j_1, j_2 \in [1, i]$ and $j_1 \neq j_2$. As $E_2(X_{i+1}, .)$ is a permutation this implies $R_{j_1} = R_{j_2} = r$. $E_1(r, .)$ being a permutation this implies $L_{j_1} = Lj_2$ as well, which is a contradiction because we have assumed previous input tuples are distinct.

Now we partition $\mathcal{S}'_{\bar{x}}(X_{i+1})$ as follows,

$$\mathcal{S}'_{\bar{x}}(X_{i+1}) = (\mathcal{S}'_{\bar{x}}(X_{i+1}) \cap \mathcal{S}^1) \cup (\mathcal{S}'_{\bar{x}}(X_{i+1}) \cap \mathcal{S}^2) \cup \cdots \cup (\mathcal{S}'_{\bar{x}}(X_{i+1}) \cap \mathcal{S}^i)$$

If we denote $|\mathcal{S}'_{\bar{x}}(X_{i+1}) \cap \mathcal{S}^j| = k_j$, and if $X_{i+1} \in \mathcal{X}_{\bar{x}}^k$ then clearly $\sum_{j=1}^{i} k_j = k$. We state this result as Lemma 3.4.

Lemma 3.4. *If X_{i+1} is new, then $\mathcal{S}'_{\bar{x}}(X_{i+1})$ is empty, and if $X_{i+1} \in \mathcal{X}_{\bar{x}}^k$ then $k = |\mathcal{S}'_{\bar{x}}(X_{i+1})| = \sum_{j=1}^{i} k_j$, where $k_j = |\mathcal{S}'_{\bar{x}}(X_{i+1}) \cap \mathcal{S}^j|$.*

Say $B_{i+1} \subseteq [1, i]$ be the set such that for all $j \in B_{i+1}$ we have $R_{i+1} = R_j$. As all the previous input tuples are distinct all x_j's are also distinct for any feasible $\bar{x} = (x_1, \cdots, x_i)$ and $j \in B_{i+1}$. Hence we get the following Lemma.

Lemma 3.5. *If $B_{i+1} \subseteq [1, i]$ be the set such that for all $j \in B_{i+1}$ we have $R_{i+1} = R_j$, then $|B_{i+1}| \leq |\mathcal{X}_{\bar{x}}|$.*

Proof.

$$|B_{i+1}| \leq \text{ number of distinct elements in any feasible tuple } \bar{x} = |\mathcal{X}_{\bar{x}}|$$

□

Now we will break the expression

$$\sum_{(s,t)\in Y^2\setminus\mathcal{OP}_i} \big|\Pr[\Psi_3(L_{i+1},R_{i+1})=(s,t)|\mathcal{V}_i\wedge\bar{X}=\bar{x}]-\Pr[\tilde{\Pi\Pi}'(L_{i+1},R_{i+1})=(s,t)|\mathcal{V}_i]\big|$$

in some separate terms so it will help us to compute the desired bound.

$$\begin{aligned}
&\sum_{(s,t)\in Y^2\setminus\mathcal{OP}_i} \big|\Pr[\Psi_3(L_{i+1},R_{i+1})=(s,t)|\mathcal{V}_i\wedge\bar{X}=\bar{x}]-\Pr[\tilde{\Pi\Pi}'(L_{i+1},R_{i+1})=(s,t)|\mathcal{V}_i]\big|\\
&\le \sum_{(s,t)\in Y^2\setminus\mathcal{OP}_i} \big|\Pr[\Psi_3(L_{i+1},R_{i+1})=(s,t)|\mathcal{V}_i\wedge\bar{X}=\bar{x}\wedge X_{i+1}\textit{ is new }]\\
&\quad-\Pr[\tilde{\Pi\Pi}'(L_{i+1},R_{i+1})=(s,t)|\mathcal{V}_i]\big|\times\Pr[X_{i+1}\textit{ is new }|\mathcal{V}_i\wedge\bar{X}=\bar{x}]\\
&\quad+\sum_{k=1}^{i}\sum_{(s,t)\in Y^2\setminus\mathcal{OP}_i} \big|\Pr[\Psi_3(L_{i+1},R_{i+1})=(s,t)|\mathcal{V}_i\wedge\bar{X}=\bar{x}\wedge X_{i+1}\textit{ is k-collision}]\\
&\quad-\Pr[\tilde{\Pi\Pi}'(L_{i+1},R_{i+1})=(s,t)|\mathcal{V}_i]\big|\times\Pr[X_{i+1}\textit{ is k-collision}|\mathcal{V}_i\wedge\bar{X}=\bar{x}]\\
&=A\times\Pr[X_{i+1}\textit{ is new }|\mathcal{V}_i\wedge\bar{X}=\bar{x}]+\sum_{k=1}^{i}C_k\times\Pr[X_{i+1}\textit{ is k-collision}|\mathcal{V}_i\wedge\bar{X}=\bar{x}]
\end{aligned}\tag{3.4}$$

Where,

$$\begin{aligned}
A=\sum_{(s,t)\in Y^2\setminus\mathcal{OP}_i} \big|\Pr[\Psi_3(L_{i+1},R_{i+1})=(s,t)|\mathcal{V}_i\wedge\bar{X}=\bar{x}\wedge X_{i+1}\textit{ is new }]\\
-\Pr[\tilde{\Pi\Pi}'(L_{i+1},R_{i+1})=(s,t)|\mathcal{V}_i]\big|
\end{aligned}\tag{3.5}$$

$$\begin{aligned}
C_k=\sum_{(s,t)\in Y^2\setminus\mathcal{OP}_i} \big|\Pr[\Psi_3(L_{i+1},R_{i+1})=(s,t)|\mathcal{V}_i\wedge\bar{X}=\bar{x}\wedge X_{i+1}\textit{ is k-collision}]\\
-\Pr[\tilde{\Pi\Pi}'(L_{i+1},R_{i+1})=(s,t)|\mathcal{V}_i]\big|
\end{aligned}\tag{3.6}$$

Now our goal is to find good upper bound of A, C_k and $\Pr[X_{i+1}\textit{ is k-collision}|\mathcal{V}_i\wedge\bar{X}=\bar{x}]$ for $k=1,\cdots,i$. Clearly,

$$\Pr[X_{i+1}\textit{ is k-collision}|\mathcal{V}_i\wedge\bar{X}=\bar{x}]=\frac{|\mathcal{X}_{\bar{x}}^k|}{|Y|-|B_{i+1}|}\tag{3.7}$$

For upper bounding A, C_k, we will use the following lemma which states the value of $\Pr[\Psi_3(L_{i+1},R_{i+1})=(s,t)|\mathcal{V}_i\wedge\bar{X}=\bar{x}\wedge X_{i+1}\textit{ is new }]$ and $\Pr[\Psi_3(L_{i+1},R_{i+1})=(s,t)|\mathcal{V}_i\wedge\bar{X}=\bar{x}\wedge X_{i+1}\textit{ is k-collision}]$ for $k=1,\cdots,i$.

Lemma 3.6. *For any $\mathcal{V}_i=((L_1,R_1,s_1,t_1),\cdots,(L_i,R_i,s_i,t_i))$ and $\bar{x}\in\mathcal{F}$, $\Psi_3(L_{i+1},R_{i+1})$ has the following conditional probability distribution*

$$\begin{aligned}
&\Pr[\Psi_3(L_{i+1},R_{i+1})=(s,t)|\mathcal{V}_i\wedge\bar{X}=\bar{x}\wedge X_{i+1}\textit{ is new }]\\
&=\begin{cases}
\frac{1}{|Y|}\times\frac{1}{|Y|} & \textit{if } (s,t)\in(Y\setminus\mathcal{S})\times Y. \quad \textit{Note } |(Y\setminus\mathcal{S})\times Y|=(|Y|-|\mathcal{S}|)|Y|\\
\frac{1}{|Y|}\times\frac{1}{|Y|-j} & \textit{if } (s,t)\in(\mathcal{S}^j\times Y)\setminus\mathcal{OP}_i. \quad \textit{Note } |(\mathcal{S}^j\times Y)\setminus\mathcal{OP}_i|=|\mathcal{S}^j|(|Y|-j)
\end{cases}
\end{aligned}$$

$$\Pr[\Psi_3(L_{i+1}, R_{i+1}) = (s,t) | \mathcal{V}_i \wedge \bar{X} = \bar{x} \wedge X_{i+1} \textit{ is k-collision}]$$
$$= \begin{cases} \frac{1}{|Y|-k} \times \frac{1}{|Y|} & \textit{if } (s,t) \in (Y \setminus \mathcal{S}) \times Y. \quad \textit{Note } |(Y \setminus \mathcal{S}) \times Y| = (|Y| - |\mathcal{S}|)|Y| \\ \frac{1}{|Y|-k} \times \frac{1}{|Y|-j} & \textit{if } (s,t) \in ((\mathcal{S}^j \setminus \mathcal{S}'_{\bar{x}}(X_{i+1})) \times Y) \setminus \mathcal{OP}_i. \quad \textit{Note } |((\mathcal{S}^j \setminus \mathcal{S}'_{\bar{x}}(X_{i+1})) \times Y) \setminus \mathcal{OP}_i| \\ & \qquad = (|\mathcal{S}^j| - k_j)(|Y| - j) \\ 0 & \textit{if } (s,t) \in (\mathcal{S}'_{\bar{x}}(X_{i+1}) \times Y) \setminus \mathcal{OP}_i. \quad \textit{Note } |(\mathcal{S}'_{\bar{x}}(X_{i+1}) \times Y) \setminus \mathcal{OP}_i| \\ & \qquad = k|Y| - \sum_{\ell=1}^{i} \ell k_\ell \end{cases}$$

where $\mathcal{OP}_i = \{(s_1,t_1), \cdots, (s_i,t_i)\}$, $\mathcal{S}^j = \{\alpha \in Y | \alpha$ *appears in* $(s_1, \cdots, s_i)$ *exactly* j *times* $\}$, $\mathcal{S} = \bigcup_{j=1}^{i} \mathcal{S}^j$, $\mathcal{S}'_{\bar{x}}(X_{i+1}) = \{\alpha \in Y | \alpha = s_j$ *and* $x_j = X_{i+1}$ *for some* $j \in [1,i]\}$ *and* $k_j = |\mathcal{S}'_{\bar{x}}(X_{i+1}) \cap \mathcal{S}^j|$.

Also we know, $\tilde{\Pi}'$ being a random permutation,

$$\Pr[\tilde{\Pi}'(L_{i+1}, R_{i+1}) = (s,t)] = \frac{1}{|Y|^2 - i}$$

for all $(s,t) \in |Y|^2 \setminus \mathcal{OP}_i$. Now we are ready to estimate A and C_k.

By Equation (3.5), we have:

$$A = (|Y| - |\mathcal{S}|)|Y| \times \Big(\frac{1}{|Y|^2 - i} - \frac{1}{|Y|^2}\Big) + \sum_{j=1}^{i} |\mathcal{S}^j|(|Y| - j) \times \Big(\frac{1}{|Y|(|Y| - j)} - \frac{1}{|Y|^2 - i}\Big)$$
$$= \frac{(|Y| - \mathcal{S})i}{|Y|(|Y|^2 - i)} + \sum_{j=1}^{i} \frac{(j|Y| - i)|\mathcal{S}^j|}{|Y|(|Y|^2 - i)}$$

By Equation (3.2) and (3.3), we have $\sum_{j=1}^{i} j|\mathcal{S}^j| = i$ and $\sum_{j=1}^{i} |\mathcal{S}^j| = |\mathcal{S}|$; this gives:

$$A = \frac{(|Y| - \mathcal{S})i}{|Y|(|Y|^2 - i)} + \frac{i|Y| - i|\mathcal{S}|}{|Y|(|Y|^2 - i)} \leq \frac{2i}{|Y|^2 - i}$$

By Equation (3.6), we have:

$$C_k = (k|Y| - \sum_{\ell=1}^{i} \ell k_\ell) \times \Big(\frac{1}{|Y|^2 - i} - 0\Big) + (|Y| - |\mathcal{S}|)|Y| \times \Big(\frac{1}{(|Y| - k)|Y|} - \frac{1}{|Y|^2 - i}\Big)$$
$$+ \sum_{j=1}^{i} (|\mathcal{S}^j| - k_j)(|Y| - j) \times \Big(\frac{1}{(|Y| - k)(|Y| - j)} - \frac{1}{|Y|^2 - i}\Big)$$
$$\leq \frac{k|Y|}{|Y|^2 - i} + \frac{(k|Y| - i)(|Y| - |\mathcal{S}|)}{(|Y| - k)(|Y|^2 - i)} + \sum_{j=1}^{i} \frac{(|\mathcal{S}^j| - k_j)(k|Y| + j|Y| - i - kj)}{(|Y| - k)(|Y|^2 - i)}$$
$$= \frac{k|Y|}{|Y|^2 - i} + \frac{(k|Y| - i)(|Y| - |\mathcal{S}|)}{(|Y| - k)(|Y|^2 - i)} + \sum_{j=1}^{i} \frac{(|\mathcal{S}^j| - k_j)(k|Y| - i)}{(|Y| - k)(|Y|^2 - i)} + \sum_{j=1}^{i} \frac{j(|\mathcal{S}^j| - k_j)}{|Y|^2 - i}$$

By Equation (3.2) and (3.3), we have $\sum_{j=1}^{i} j|\mathcal{S}^j| = i$ and $\sum_{j=1}^{i} |\mathcal{S}^j| = |\mathcal{S}|$, and by Lemma 3.4, we have $\sum_{j=1}^{i} k_j = k$; this gives:

$$\begin{aligned} C_k &\leq \frac{k|Y|}{|Y|^2 - i} + \frac{(k|Y| - i)(|Y| - |\mathcal{S}|)}{(|Y| - k)(|Y|^2 - i)} + \frac{(k|Y| - i)(|\mathcal{S}| - k)}{(|Y| - k)(|Y|^2 - i)} + \frac{i}{|Y|^2 - i} \\ &= \frac{k|Y|}{|Y|^2 - i} + \frac{k|Y| - i}{|Y|^2 - i} + \frac{i}{|Y|^2 - i} \\ &= \frac{2k|Y|}{|Y|^2 - i} \end{aligned}$$

Now putting the upper bounds of A and C_k in Equation (3.4) we get,

$$\begin{aligned} &\sum_{(s,t)\in Y^2 \setminus \mathcal{OP}_i} |\Pr[\Psi_3(L_{i+1}, R_{i+1}) = (s,t) | \mathcal{V}_i \wedge \bar{X} = \bar{x}] - \Pr[\tilde{\Pi}'(L_{i+1}, R_{i+1}) = (s,t) | \mathcal{V}_i]| \\ &\leq \frac{2i}{|Y|^2 - i} \times \Pr[X_{i+1} \textit{ is new } | \mathcal{V}_i \wedge \bar{X} = \bar{x}] \\ &\qquad + \sum_{k=1}^{i} \frac{2k|Y|}{|Y|^2 - i} \times \Pr[X_{i+1} \textit{ is k-collision} | \mathcal{V}_i \wedge \bar{X} = \bar{x}] \\ &= \frac{2i}{|Y|^2 - i} \big(\Pr[X_{i+1} \textit{ is new } | \mathcal{V}_i \wedge \bar{X} = \bar{x}] + \sum_{k=1}^{i} \Pr[X_{i+1} \textit{ is k-collision} | \mathcal{V}_i \wedge \bar{X} = \bar{x}] \big) \\ &\quad + \sum_{k=1}^{i} \frac{2(k|Y| - i)}{|Y|^2 - i} \times \Pr[X_{i+1} \textit{ is k-collision} | \mathcal{V}_i \wedge \bar{X} = \bar{x}] \\ &= \frac{2i}{|Y|^2 - i} + \sum_{k=1}^{i} \frac{2(k|Y| - i)}{|Y|^2 - i} \times \frac{|\mathcal{X}_{\bar{x}}^k|}{|Y| - |B_{i+1}|} \qquad \textit{By Equation (3.7)} \\ &= \frac{2i}{|Y|^2 - i} + \frac{2|Y| \sum_{k=1}^{i} k|\mathcal{X}_{\bar{x}}^k| - 2i \sum_{k=1}^{i} |\mathcal{X}_{\bar{x}}^k|}{(|Y|^2 - i)(|Y| - |B_{i+1}|)} \\ &= \frac{2i}{|Y|^2 - i} + \frac{2i}{|Y|^2 - i} \times \frac{|Y| - |\mathcal{X}_{\bar{x}}|}{|Y| - |B_{i+1}|} \\ &\qquad \textit{By Equation (3.1) \& (3.3)} \, , \sum_{k=1}^{i} k|\mathcal{X}_{\bar{x}}^k| = i \textit{ and } \sum_{k=1}^{i} |\mathcal{X}_{\bar{x}}^k| = |\mathcal{X}_{\bar{x}}| \\ &\leq \frac{4i}{|Y|^2 - i} \qquad \textit{By Lemma 3.5 we have } |\mathcal{X}_{\bar{x}}| \geq |B_{i+1}| \end{aligned}$$

□

Proof of Lemma 3.6

Note $\Psi_3(L_{i+1}, R_{i+1}) = (s,t)$ actually means,

$$\begin{aligned} s &= E_2(X_{i+1}, R_{i+1}) \\ t &= E_3(s, X_{i+1}) \end{aligned}$$

We know E_2, E_3 are random permutations. That means if at some point of time, for some particular key K, $(I_1, O_1), \cdots, (I_\ell, O_\ell)$ input-output pairs have already been fixed for the random permutation $E_2(K, .)$, then at the next invokation of $E_2(K, .)$,

$$\Pr[E_2(K, x) = y] = \frac{1}{|Y| - \ell}$$

for all $x \in Y \setminus \{I_1, \cdots, I_\ell\}$ and $y \in Y \setminus \{O_1, \cdots, O_\ell\}$. The same is true for E_3 random permutation.
Hence if X_{i+1} is new, then

$$\Pr[E_2(X_{i+1}, R_{i+1}) = s] = \frac{1}{|Y|}$$

for all $s \in Y$.
If X_{i+1} is k-collision, then

$$\Pr[E_2(X_{i+1}, R_{i+1}) = s] = \frac{1}{|Y| - k}$$

for all $s \in Y \setminus \mathcal{S}'_{\bar{x}}(X_{i+1})$. And

$$\Pr[E_2(X_{i+1}, R_{i+1}) = s] = 0$$

for all $s \in \mathcal{S}'_{\bar{x}}(X_{i+1})$, because otherwise we have a duplicate query.
Similarly, if $s \in Y \setminus \mathcal{S}$, then

$$\Pr[E_3(s, X_{i+1}) = t] = \frac{1}{|Y|}$$

for all $t \in Y$.
And if $s \in \mathcal{S}^j$, then

$$\Pr[E_3(s, X_{i+1}) = t] = \frac{1}{|Y| - j}$$

for all $t \in Y \setminus$ *Set of t values corresponding to s in $\mathcal{V}_i$*. Using the above probability values it is easy to see why Lemma 3.6 holds. □

Theorem 3.6. *The tweakable block-cipher $\tilde{\Psi}_3$ is a (t', q, ε')-secure tweakable block-cipher, if $\tilde{E}_1$, $\tilde{E}_2$ and $\tilde{E}_3$ are all (t, q, ε)-secure tweakable block-ciphers, where $\varepsilon' = 3 \cdot \varepsilon + q^2/2^{2n}$ and $t' = t - \mathcal{O}(qn)$.*

Proof. Theorem 3.5 and the following sequence of games completes the proof of Theorem 3.6. We denote S_i the event that the distinguisher outputs 1 in $\texttt{Game}_i$.

$\texttt{Game}_0$: the queries are answered using $\tilde{\Psi}_3$, as illustrated in Fig. 3.5.

$\texttt{Game}_1$: we replace the tweakable block-ciphers E_1, E_2, E_3 by 3 independent family of random permutations. From an attacker against $\tilde{\Psi}_3$ running in time t', we can construct an attacker against E_1, E_2 or E_3 running in time at most:

$$t = t' + \mathcal{O}(qn)$$

Since by assumption E_1, E_2 and E_3 are all (t, q, ε)-secure, we must have:

$$|\Pr[S_1] - \Pr[S_0]| \leq 3 \cdot \varepsilon$$

$\texttt{Game}_2$: the adversary interacts with a family of random permutation $\tilde{\Pi}'$. By Theorem 3.5 we must have:

$$|\Pr[S_2] - \Pr[S_1]| \leq \frac{q^2}{2^{2n}}$$

□

One drawback of our construction is that it shrinks the tweak size from ω bits to $\omega - n$ bits. We show a simple construction that extends the tweak size, using a keyed universal hash function; this construction can be of independent interest.

Definition 3.3. *A family $\mathcal{H}$ of functions with signature $\{0,1\}^{\omega'} \to \{0,1\}^{\omega}$ is said to be ε-almost universal if $\Pr_h[h(x) = h(y)] \leq \varepsilon$ for all $x \neq y$, where the probability is taken over h chosen uniformly at random from $\mathcal{H}$.*

Let $\tilde{E}$ be a tweakable block-cipher with tweak in $\{0,1\}^{\omega}$. Given a family $\mathcal{H}$ of hash functions h with signature $\{0,1\}^{\omega'} \to \{0,1\}^{\omega}$ and $\omega' > \omega$, our tweakable block-cipher $\tilde{E}$ with extended tweak length ω' is defined as:

$$\tilde{E}'((K,h), W', m) = \tilde{E}(K, h(W'), m)$$

Theorem 3.7. *The tweakable block cipher $\tilde{E}'$ is a (q, t', ε')-secure tweakable block cipher if $\tilde{E}$ is a (q, t, ε_1)-secure tweakable block cipher and the hash function family $\mathcal{H}$ is ε_2-almost universal, with $\varepsilon' = \varepsilon_1 + q^2 \cdot \varepsilon_2$ and $t' = t - \mathcal{O}(q)$.*

Proof. We consider a (q, t', ε')-adversary $\mathcal{A}'$ against our construction $\tilde{E}'$. We must describe a (q, t, ε_1)-adversary $\mathcal{A}$ against the original tweakable block cipher $\tilde{E}$. Our adversary $\mathcal{A}$ has oracle access to F and F^{-1}, where either $F = \tilde{E}(K, \cdot, \cdot)$ or $F = \tilde{\Pi}(\cdot, \cdot)$; it must output a bit γ, representing its guess as to whether $F = \tilde{E}(K, \cdot, \cdot)$ or $F = \tilde{\Pi}(\cdot, \cdot)$.

We first generate a random $h \in \mathcal{H}$. When $\mathcal{A}'$ queries for $F'(W', m)$, we compute $h(W')$ and return $F(h(W'), m)$, and similarly for a F'^{-1} query. Eventually, $\mathcal{A}'$ outputs a bit γ, which is returned by our adversary $\mathcal{A}$.

When $F = \tilde{E}(K, \cdot, \cdot)$, we have that adversary $\mathcal{A}'$ interacts with $F' = \tilde{E}'((K,h), \cdot, \cdot)$, exactly as in the security definition, which gives:

$$\Pr[\gamma = 1 | F = \tilde{E}(K, \cdot, \cdot)] = \Pr[\gamma = 1 | F' = \tilde{E}'((K,h), \cdot, \cdot)]$$

When $F = \tilde{\Pi}(\cdot, \cdot)$ we must show that the view of adversary $\mathcal{A}'$ is statistically close to that of $\mathcal{A}'$ in the original security definition. In the security definition, $\mathcal{A}'$ interacts with a family $\tilde{\Pi}'$ of independent random permutation parametrised with W'. Here instead the adversary $\mathcal{A}'$ interacts with $\tilde{\Pi}(h(\cdot), \cdot)$. The key observation is that if no collision occurs for h, then the distribution seen by $\mathcal{A}'$ is exactly the same as the one obtained from $\tilde{\Pi}'$. Let denote by $\texttt{Bad}$ the event that such collision

occurs; since $\mathcal{H}$ is a family of ε_2-almost universal hash functions and there are at most q queries, we have:

$$\Pr[\mathsf{Bad}] \leq q^2 \cdot \varepsilon_2$$

Moreover we obtain:

$$\Pr[\gamma = 1 | F = \tilde{\Pi} \wedge \neg \mathsf{Bad}] = \Pr[\gamma = 1 | F' = \tilde{\Pi}']$$

which gives:

$$|\Pr[\gamma = 1 | F = \tilde{\Pi}] - \Pr[\gamma = 1 | F' = \tilde{\Pi}']| \leq \Pr[\mathsf{Bad}] \leq q^2 \cdot \varepsilon_2$$

Eventually denoting:

$$\begin{aligned} \delta &= |\Pr[\gamma = 1 | F = \tilde{E}(K, \cdot, \cdot)] - \Pr[\gamma = 1 | F = \tilde{\Pi}]| \\ \delta' &= |\Pr[\gamma = 1 | F' = \tilde{E}'((K, h), \cdot, \cdot)] - \Pr[\gamma = 1 | F' = \tilde{\Pi}']| \end{aligned}$$

we obtain:

$$\delta' \leq \delta + q^2 \cdot \varepsilon_2$$

Since by assumption $\delta \leq \varepsilon_1$, we obtain $\delta' \leq \varepsilon_1 + q^2 \cdot \varepsilon_2$; therefore we can take:

$$\varepsilon' = \varepsilon_1 + q^2 \cdot \varepsilon_2$$

□

We note that many efficient constructions of universal hash function families are known, with $\varepsilon_2 \simeq 2^{-\omega}$. Therefore the new tweakable block-cipher can have the same level of security as the original one, up to the birthday bound for the tweak, *i.e.* for $q \leq 2^{\omega/2}$.

3.7 Conclusion

We have described the first domain extender for ideal ciphers, *i.e.* we have showed a construction that is indifferentiable from a $2n$-bit ideal cipher, given a n-bit ideal cipher. Our construction is based on a 3-round Feistel, and is more efficient and more secure than first building a n-bit random oracle from a n-bit ideal cipher (as in [CDMP05]) and then a $2n$-bit ideal cipher from a n-bit random oracle (as in [CPS08]). We have also shown that in the standard model, our construction with 2 rounds enables to get a $2n$-bit tweakable block-cipher from a n-bit tweakable block-cipher and that with 3 rounds we get a security guarantee beyond the birthday paradox.

Chapter 4

On the Public Indifferentiability and Correlation Intractability of the 6-Round Feistel Construction

We show that the Feistel construction with six rounds and random round functions is *publicly* indifferentiable from a random invertible permutation (a result that is not known to hold for full indifferentiability). Public indifferentiability (*pub-indifferentiability* for short) is a variant of indifferentiability introduced by Yoneyama *et al.* [YMO09] and Dodis *et al.* [DRS09] where the simulator knows all queries made by the distinguisher to the primitive it tries to simulate, and is useful to argue the security of cryptosystems where all the queries to the ideal primitive are public (as *e.g.* in many digital signature schemes). To prove the result, we introduce a new and simpler variant of indifferentiability, that we call sequential indifferentiability (*seq-indifferentiability* for short) and show that this notion is in fact equivalent to pub-indifferentiability for stateless ideal primitives. We then prove that the 6-round Feistel construction is seq-indifferentiable from a random invertible permutation. We also observe that sequential indifferentiability implies correlation intractability, so that the Feistel construction with six rounds and random round functions yields a correlation intractable invertible permutation, a notion we define analogously to correlation intractable functions introduced by Canetti *et al.* [CGH98]. We also show the 3-round domain extender construction from the previous section (Section 3) actually guarantees beyond birthday security against sequential indifferentiability (which in turn implies public indifferentiability) attackers. This is a joint work with Jacques Patarin and Yannick Seurin [MPS12].

Contents

4.1 Introduction

The Feistel construction with public round functions. The Feistel construction turns a function F from n-bit strings to n-bit strings into an (efficiently invertible) permutation on $2n$-bit strings. It is computed as $\Psi^F(L, R) = (R, L \oplus F(R))$. In their seminal paper [LR88] which triggered a lot of subsequent work [Mau92, NR99, Pat90, Pat91, Pat98, Pat03, Pat04, Vau03], Luby and Rackoff showed that three (resp. four) rounds of the Feistel construction, with independent pseudorandom functions in each round, yields a pseudorandom permutation (resp. strong pseudorandom permutation). The core of this result is in fact purely information-theoretic [Mau92], meaning that the Feistel construction with three (resp. four) rounds and random round functions is indistinguishable from a random permutation (resp. an invertible random permutation) by any *computationally unbounded* distinguisher limited to a *polynomial number of oracle queries*. The Luby-Rackoff theorem crucially relies on the secrecy of the round functions. A few papers studied what happens when the round functions are made public. In particular, Ramzan and Reyzin [RR00] have shown that the Feistel construction with four rounds remains strongly pseudorandom even when the distinguisher has oracle access to the two middle round functions (but not to the first or the fourth round function). Dodis and Puniya [DP07] have studied various properties of the Feistel construction (unpredictability, pseudorandomness) when all intermediate round values of the Feistel computation are leaked to the adversary and shown that in that case a super-logarithmic number of rounds was necessary and sufficient for the property to be inherited by the Feistel construction from the round functions.

Indifferentiability of the Feistel construction. Coron *et al.* [CDMP05] showed that a number of variants of the Merkle-Damgård construction [Dam89, Mer89],

used with an ideal cipher in Davies-Meyer mode [PGV93, BRS02], are indifferentiable from a random oracle. Hence, it is possible to securely instantiate a random oracle in the ideal cipher model. A natural question is whether the other direction holds, namely whether there is a construction using a random oracle that securely implements a random invertible permutation.[1] Given its numerous cryptographic properties, the Feistel construction (with public random round functions) appears as an obvious candidate for this task. Again, this question can be rigorously formulated in the indifferentiability framework: namely, is the Feistel construction with sufficiently many rounds, and public random round functions, indifferentiable from a random invertible permutation? Dodis and Puniya [DP06] considered the problem in the so-called *honest-but-curious* model, where the distinguisher only sees the queries made by the Feistel construction to the random round functions, but is not allowed to make arbitrary queries to the round functions. In this setting, they showed that a super-logarithmic number of rounds is sufficient to securely realize a random invertible permutation. However, since full indifferentiability is not implied in general by indifferentiability in the honest-but-curious model (these two notions are in fact incomparable [CPS08]), they were not able to conclude in the general setting. Coron, Patarin, and Seurin [CPS08] gave a first proof that the Feistel construction with six rounds is indifferentiable from a random invertible permutation. The proof was rather involved, and Künzler [KÖ9] later found a distinguishing attack against the simulator given in [CPS08], therefore invalidating the indifferentiability proof.[2] Only recently, Holenstein *et al.* [HKT11] gave a new proof that the Feistel construction with *fourteen* rounds is indifferentiable from a random invertible permutation, which was inspired from a previous proof for ten rounds that appeared in the PhD thesis of Seurin [Seu09] but had some gaps.

Public indifferentiability. Yoneyama *et al.* [YMO09] and Dodis *et al.* [DRS09] independently realized that indifferentiability was sometimes stronger than needed to argue security of cryptosystems. In particular, when all queries made to the ideal primitive are public (like in many digital signature schemes such as FDH [BR93], probabilistic FDH [Cor02], PSS [BR96]..., where all queries to the hash function can be revealed to the attacker without affecting the security), the weaker notion of *public* indifferentiability is sufficient. [YMO09, DRS09] were both concerned with indifferentiability from a random oracle and respectively called this notion *leaky random oracle* and *public-use random oracle*. Public indifferentiability is defined similarly to indifferentiability, but the task of the simulator is made easier by letting it know all queries made by the distinguisher to the ideal primitive $\boldsymbol{G}$.

Correlation intractability. Correlation intractability was introduced by Canetti *et al.* [CGH98] as an attempt to capture as many security properties of the random oracle as possible. A family of functions is said to be correlation intractable if for a random function of the family it is hard to find a sequence of inputs that together with their image satisfy a relation that would be hard to satisfy for a uniformly ran-

[1] Such a construction easily implies a secure ideal cipher by simply prepending the key of the block cipher to the input of each random oracle queries.

[2] We stress that this does not mean that the 6-round Feistel construction is not indifferentiable from a random invertible permutation, but only that no one is able to give a proof at the moment.

dom function (a so-called *evasive* relation). Correlation intractability in particular implies collision resistance, pre-image resistance and many other security properties usually required for cryptographic hash functions. Unfortunately, Canetti *et al.* also showed that in the standard model, no correlation intractable hash function family exists. A consequence of this non-existence result is that there are cryptosystems that are secure in the random oracle model, but insecure when the random oracle is instantiated by any function family. Though correlation intractability was primarily defined in the standard model, it is easily transformable to idealized models. As we will see our result establishes a connection between correlation intractability and public indifferentiability.

Contributions of this work. We define a new and weaker notion of indifferentiability that we call *sequential* indifferentiability (*seq-indifferentiability* for short). This new definition only restricts *the order* in which the distinguisher can query the two oracles it is granted access to: it can first query the primitive $\boldsymbol{F}$ (or the simulator $\mathcal{S}$), and then the construction $\mathcal{C}^{\boldsymbol{F}}$ (or the ideal primitive $\boldsymbol{G}$), but not $\boldsymbol{F}/\mathcal{S}$ again. We show that when the ideal primitive $\boldsymbol{G}$ is stateless (which is the most usual case), this notion is equivalent to *public* indifferentiability introduced by [DRS09, YMO09] where all queries to the primitive $\boldsymbol{G}$ are public. However the seq-indifferentiability notion has the advantage of being simpler and easier to use in proofs. This simple restriction on the queries of the distinguisher enables to give a relatively simple proof that the 6-round Feistel construction with random round functions is seq-indifferentiable (and hence also publicly indifferentiable) from a random invertible permutation, a result whose analogue for full indifferentiability seems out of reach at the moment. Our result in particular implies that any scheme proven secure in the random invertible permutation model or the ideal cipher model and where all queries to the ideal primitive can be made public without affecting the security (*e.g.* signature schemes like OPSSR [Gra02] and subsequent variants [KW03, CMPP05]) remains secure in the random oracle model when using a 6-round Feistel construction (while the best generic replacement previously to our work was the 14-round Feistel construction [HKT11]).

Though weaker than full indifferentiability, we also show that seq-indifferentiability is still sufficiently strong to imply correlation intractability. In particular, our result shows that the 6-round Feistel construction with random round functions yields a correlation intractable invertible permutation (we note that previous observations [CPS08] already implied that the 5-round Feistel construction fails to provide a correlation intractable invertible permutation). We discuss the implications of this result for chosen-key and known-key attacks on block ciphers [KR07].

On a slightly different topic, we also analyze the Feistel-like domain extension construction for ideal ciphers proposed by Coron *et al.* [CDMS10] and show that in the seq-indifferentiability model one can obtain a security bound beyond the birthday barrier.

Open problems. The most challenging open question is of course whether the 6-round Feistel construction is fully indifferentiable from a random invertible permutation, and if not, what is the minimal number of rounds needed to achieve this property. We hope that our result will constitute a first step towards a finer un-

derstanding of this question. In particular, our result implies that if the 6-round Feistel construction is *not* fully indifferentiable from a random invertible permutation, then this cannot be shown by proving that it is not correlation intractable as was done for five rounds. Another interesting problem is to weaken the assumptions on the round functions and see which property would continue to hold: *e.g.* is the 6-round Feistel construction with correlation intractable round functions still a correlation intractable invertible permutation? A related question is whether our result could be a first step towards proposing plausible constructions of (restricted) correlation intractable function families in the standard model, a question left open by [CGH98, Section 5.1].

Organization. In Section 4.2, we start by giving the definition of sequential indifferentiability and prove that it is equivalent to public indifferentiability for stateless ideal primitives. In Section 4.5, we prove the main result of this section, namely that the 6-round Feistel construction is sequentially (and hence publicly) indifferentiable from a random invertible permutation. In Section 4.6, we apply this result to prove the correlation intractability of the 6-round Feistel construction.

4.2 Preliminaries

Ideal primitives. We recall the notion of ideal primitives from Chapter 2. Given two sets $\mathtt{Dom} \subset \{0,1\}^*$ and $\mathtt{Rng} \subset \{0,1\}^*$, we denote $\mathcal{F}(\mathtt{Dom}, \mathtt{Rng})$ the set of all functions from $\mathtt{Dom}$ to $\mathtt{Rng}$. A primitive $\mathbb{G}$ is a sequence $\mathbb{G} = (\mathtt{Dom}_n, \mathtt{Rng}_n, \mathbb{G}_n)_{n\in\mathbb{N}}$ where $\mathbb{G}_n \subset \mathcal{F}(\mathtt{Dom}_n, \mathtt{Rng}_n)$. The ideal primitive $\boldsymbol{G}$ associated with $\mathbb{G}$ is the sequence of random variables $(\boldsymbol{G}_n)_{n\in\mathbb{N}}$ where $\boldsymbol{G}_n$ is uniformly distributed over $\mathbb{G}_n$. We will often adopt the lazy sampling view [BR06] to describe ideal primitives queried as oracles.

A random function $\boldsymbol{F} = (\boldsymbol{F}_n)_{n\in\mathbb{N}}$ is the ideal primitive associated to the set of all functions from $\{0,1\}^n$ to $\{0,1\}^n$. Queried as an oracle it returns a uniformly random string in $\{0,1\}^n$ if x was never queried, or the same answer as before if x was previously queried.

A random invertible permutation $\boldsymbol{P} = (\boldsymbol{P}_n)_{n\in\mathbb{N}}$ is the ideal primitive associated with the sequence $\mathbb{P} = (\mathtt{Dom}_n, \mathtt{Rng}_n, \mathbb{P}_n)_{n\in\mathbb{N}}$ where $\mathtt{Dom}_n = \{0,1\} \times \{0,1\}^n$, $\mathtt{Rng}_n = \{0,1\}^n$, and $\mathbb{P}_n$ is the set of functions P such that $x \mapsto P(0,x)$ is a permutation of $\{0,1\}^n$, and $y \mapsto P(1,y)$ its inverse. Queries of the form $(0,x)$ and $(1,y)$ will be called respectively *forward* and *backward* queries. In the lazy sampling point of view, $\boldsymbol{P}_n$ keeps two lists L_x and L_y of forward and backward queries whose image is already defined together with an invertible mapping from L_x to L_y. Upon receiving a forward query $(0,x)$ such that $x \notin L_x$ it returns an answer y uniformly random over $\{0,1\}^n \setminus L_y$, and adds x to L_x and y to L_y and updates the mapping (and reciprocally for a backward query $(1,y)$). Later, we will occasionally refer to L_x and L_y as the *history* of the random invertible permutation. An ideal cipher $\boldsymbol{E} = (\boldsymbol{E}_n)$ takes an additional input, the key, of length $\ell(n)$, and for each key $k \in \{0,1\}^{\ell(n)}$, $\boldsymbol{E}_n(k,\cdot)$ is an independent random invertible permutation over $\{0,1\}^n$.

A two-sided random function on $\{0,1\}^n$, denoted $\boldsymbol{R}_n$, is very similar to a random

invertible permutation. It also keeps to lists L_x and L_y together with an invertible mapping from L_x to L_y. However when receiving a forward query $(0, x)$ such that $x \notin L_x$ or a backward query $(1, y)$ such that $y \notin L_y$, it returns a *uniformly random* answer in $\{0,1\}^n$. In case a collision happens, the previous image or pre-image is removed from L_y or L_x and the mapping is updated accordingly. Note that a two-sided random function is stateful: it may return different answers to the same query (however at any time it defines an invertible mapping from L_x to L_y). A two-sided random function is statistically indistinguishable from a random invertible permutation: the so called PRF/PRP switching lemma [BR06] establishes[3] that an oracle machine making at most q oracle queries can distinguish $\boldsymbol{P}_n$ from $\boldsymbol{R}_n$ with advantage at most $q^2/2^{n+1}$.

In the following, we omit the subscripts when the domain and the range of an ideal primitive are clear from the context. A *construction* will simply be a Turing machine having oracle access to an ideal primitive and implementing another given primitive. The main construction we will consider in this work is the Feistel construction.

The Feistel construction. Given a function $F : \{0,1\}^n \to \{0,1\}^n$, the basic (1-round) Feistel construction is the permutation on $\{0,1\}^{2n}$ defined by $\Psi^F(L, R) = (R, L \oplus F(R))$. Its inverse is computed by $(\Psi^F)^{-1}(S, T) = (T \oplus F(S), S)$. (Here L, R, S, and T are n-bit strings). The k-round Feistel construction associated to round functions $(F_1, \ldots, F_k)$ takes inputs $x \in \{0,1\} \times \{0,1\}^{2n}$ and is defined by:

$$\Psi_k^{(F_1,\ldots,F_k)}(0,(L,R)) = \Psi^{F_k} \circ \cdots \circ \Psi^{F_1}(L,R)$$
$$\Psi_k^{(F_1,\ldots,F_k)}(1,(S,T)) = \left(\Psi^{F_1}\right)^{-1} \circ \cdots \circ \left(\Psi^{F_k}\right)^{-1}(S,T) \ .$$

Notations used for denoting the intermediate round values for the 6-round Feistel construction are given in Figure 4.1. In the following, when considering the Feistel construction using k independent random functions, we will simply note $\boldsymbol{F} = (\boldsymbol{F}_1, \ldots, \boldsymbol{F}_k)$ this tuple of functions and $\Psi_k^{\boldsymbol{F}} = \Psi_k^{(\boldsymbol{F}_1,\ldots,\boldsymbol{F}_k)}$.

4.3 Sequential Indifferentiability

Indifferentiability was originally formulated within the formalism of *random systems* [Mau02]. We adopt here the simpler formulation using interactive Turing machines as in [CDMP05]. We first recall the classical definition of indifferentiability [MRH04]. For this, we slightly change the way one usually measure the cost of queries of a distinguisher (this will make our results simpler to express). Given a distinguisher $\mathcal{D}$, the *total oracle queries cost* of $\mathcal{D}$ is the number of queries received by the oracle $\boldsymbol{F}$ when $\mathcal{D}$ interacts with $(\mathcal{C}^{\boldsymbol{F}}, \boldsymbol{F})$. Hence this is the sum of the number of direct queries of $\mathcal{D}$ to $\boldsymbol{F}$ and the number of queries made by $\mathcal{C}$ to $\boldsymbol{F}$ to answer $\mathcal{D}$'s queries. We recall the definition of indifferentiability (Definition 3.2) from Chapter 2.

[3] Strictly speaking, the result is proven in [BR06] for one-sided functions and permutations, but the proof can be straightforwardly adapted to two-sided primitives.

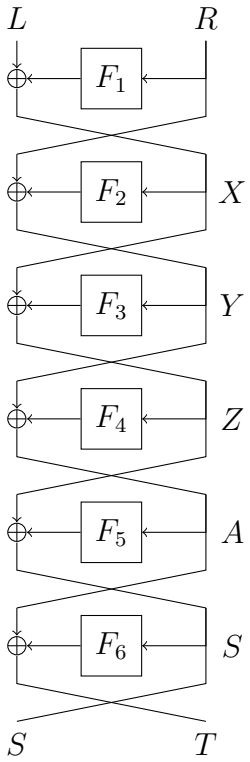

Figure 4.1: Notations used for the 6-round Feistel construction.

Definition 4.1 (Indifferentiability)**.** *A construction $\mathcal{C}$ with oracle access to an ideal primitive $\boldsymbol{F}$ is said to be (q, t, σ, ϵ)-indifferentiable from an ideal primitive $\boldsymbol{G}$ if there exists an oracle Interactive Turing Machine (ITM) $\mathcal{S}$ such that for any distinguisher $\mathcal{D}$ of total oracle queries cost at most q, $\mathcal{S}$ runs at most at time t making at most σ oracle queries, and the following holds:*

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

$\mathcal{C}^{\boldsymbol{F}}$ is simply said to be indifferentiable from $\boldsymbol{G}$ if for any q polynomial in the security parameter n, the above definition is fulfilled with a σ bounded above by a polynomial in the security parameter n and some ϵ which is a negligible function of n.

In order to define our new notion of indifferentiability, we will consider a restricted class of distinguisher, called *sequential distinguisher*, which can only make queries in a specific order. Such a distinguisher first queries the primitive $\boldsymbol{F}$ (or the simulator $\mathcal{S}$) as it wishes, and then the construction $\mathcal{C}^{\boldsymbol{F}}$ (or the primitive $\boldsymbol{G}$) as it wishes, but after its first query to $\mathcal{C}^{\boldsymbol{F}}$ or $\boldsymbol{G}$, it cannot query $\mathcal{S}$ or $\boldsymbol{F}$ again. Sequential indifferentiability (*seq-indifferentiability* for short) is defined relatively to such distinguishers (see also Figure 4.2).

Definition 4.2 (Seq-indifferentiability)**.** *A construction $\mathcal{C}$ with oracle access to an ideal primitive $\boldsymbol{F}$ is said to be (q, t, σ, ϵ)-seq-indifferentiable from an ideal primitive*

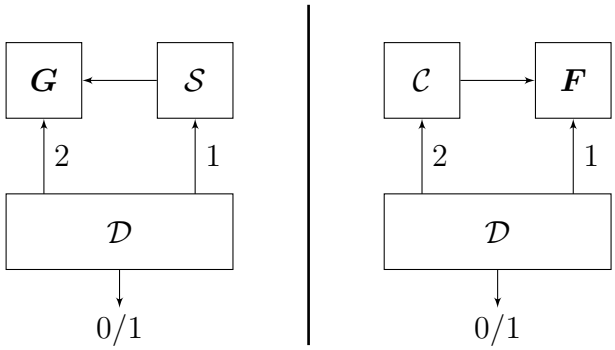

Figure 4.2: The sequential indifferentiability notion. The numbers next to query arrows indicate in which order the distinguisher accesses both oracles. After its first query to the left oracle, the distinguisher cannot query the right oracle any more.

$\boldsymbol{G}$ if Definition 4.1 is fulfilled when $\mathcal{D}$ ranges over the class of sequential distinguishers.

Full indifferentiability obviously implies seq-indifferentiability. Yoneyama *et al.* [YMO09] and Dodis *et al.* [DRS09] have introduced another weakened notion of indifferentiability, where the primitive $\boldsymbol{G}$ is only queried on *public* inputs, that we call here public indifferentiability (*pub-indifferentiability* for short). This can be formalized as follows: given an ideal primitive $\boldsymbol{G}$, we define the augmented ideal primitive $\overline{\boldsymbol{G}}$ as the primitive exposing two interfaces: the first (regular) one is the same as $\boldsymbol{G}$, and the second is an interface `Reveal` that, when queried, returns the ordered sequence of all (regular) queries and corresponding answers made so far by any party to the regular interface. The second interface can only be used by the simulator, not by the distinguisher. We recall the formal definition of public indifferentiability (Definition 2.4) from Chapter 2.

Definition 4.3 (Pub-indifferentiability)**.** *A construction $\mathcal{C}$ with oracle access to an ideal primitive $\boldsymbol{F}$ is said to be (q, t, σ, ϵ)-pub-indifferentiable from an ideal primitive $\boldsymbol{G}$ if there exists an oracle ITM $\mathcal{S}$ such that for any distinguisher $\mathcal{D}$ of total oracle queries cost at most q, $\mathcal{S}$ runs at most at time t and makes at most σ oracle queries, and the following holds:*

$$\left|\Pr\left[\mathcal{D}^{\boldsymbol{G},\mathcal{S}^{\overline{\boldsymbol{G}}}} = 1\right] - \Pr\left[\mathcal{D}^{\mathcal{C}^{\boldsymbol{F}},\boldsymbol{F}} = 1\right]\right| \leq \epsilon \ .$$

As explained in [DRS09], the composition theorem of [MRH04] still holds with pub-indifferentiability for cryptosystems where all messages queried to $\boldsymbol{G}$ can be inferred from the adversary's query during the security experiment.

Clearly, pub-indifferentiability implies seq-indifferentiability. Indeed, since after its first query to $\boldsymbol{G}$ a sequential distinguisher never queries the simulator again, the interface `Reveal` is of no use to the simulator. A less trivial result is that seq-

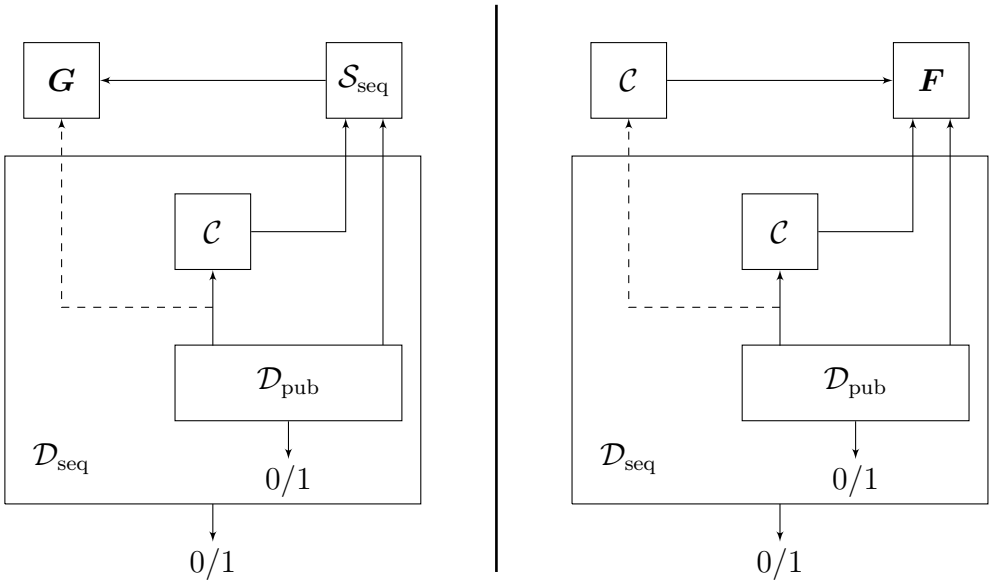

Figure 4.3: Illustration of the proof of Theorem 4.1. The dashed arrow means that $\mathcal{D}_{\text{seq}}$ makes the corresponding queries once $\mathcal{D}_{\text{pub}}$ has returned and compares the answers with the one it computed with $\mathcal{C}$.

indifferentiability implies pub-indifferentiability *for stateless*[4] *ideal primitives* $\boldsymbol{G}$, thus making seq- and pub-indifferentiability equivalent notions in that case.

Theorem 4.1. *Let $\mathcal{C}$ be a construction with oracle access to some ideal primitive $\boldsymbol{F}$. If $\mathcal{C}^{\boldsymbol{F}}$ is $(2q, t, \sigma, \epsilon)$-seq-indifferentiable from a* stateless *ideal primitive $\boldsymbol{G}$, then $\mathcal{C}^{\boldsymbol{F}}$ is $(q, t', \sigma + q, \epsilon)$-pub-indifferentiable from $\boldsymbol{G}$, where $t = t' + O(qn)$.*

Proof. Assume that $\mathcal{C}^{\boldsymbol{F}}$ is $(2q, t, \sigma, \epsilon)$-seq-indifferentiable from $\boldsymbol{G}$, and let $\mathcal{S}_{\text{seq}}$ be the simulator for seq-indifferentiability. We define a simulator $\mathcal{S}_{\text{pub}}$ for pub-indifferentiability as follows. $\mathcal{S}_{\text{pub}}$ runs $\mathcal{S}_{\text{seq}}$, transparently relaying queries of $\mathcal{S}_{\text{seq}}$ to $\boldsymbol{G}$ to the regular interface of $\overline{\boldsymbol{G}}$. Each time $\mathcal{S}_{\text{pub}}$ receives a F-query y from the distinguisher, it makes a call to `Reveal`, getting a sequence $(x_1, \ldots, x_m)$ of G-queries that have been made by the distinguisher so far ($\mathcal{S}_{\text{pub}}$ considers only *fresh* G-queries, *i.e.* G-queries that have not been returned by a previous query to `Reveal`). For each $i = 1$ to m, $\mathcal{S}_{\text{pub}}$ makes all F-queries needed to compute $\mathcal{C}^F(x_i)$ to $\mathcal{S}_{\text{seq}}$. Finally, it makes the F-query y to $\mathcal{S}_{\text{seq}}$ and returns the corresponding answer.

Let $\mathcal{D}_{\text{pub}}$ be a distinguisher for the pub-indifferentiability game of total oracle queries cost at most q. We have to bound the absolute difference between the probabilities that $\mathcal{D}_{\text{pub}}$ outputs 1 when interacting with $(\boldsymbol{G}, \mathcal{S}_{\text{pub}}^{\overline{\boldsymbol{G}}})$ and $(\mathcal{C}^{\boldsymbol{F}}, \boldsymbol{F})$. For

[4]By stateless we mean that the answer of $\boldsymbol{G}$ to any query only depends on the query and the randomness of $\boldsymbol{G}$ and not on any additional state information. In particular, for fixed randomness, $\boldsymbol{G}$ always returns the same answer to a given query.

this, we assume *wlog* that

$$\Pr\left[\mathcal{D}_{\text{pub}}^{\boldsymbol{G},\mathcal{S}_{\text{pub}}^{\overline{\boldsymbol{G}}}}=1\right] \geq \Pr\left[\mathcal{D}_{\text{pub}}^{\mathcal{C}^{\boldsymbol{F}},\boldsymbol{F}}=1\right] \quad . \tag{4.1}$$

We consider the following sequential distinguisher $\mathcal{D}_{\text{seq}}$ interacting with a pair of oracles (G, F) which can be either $(\boldsymbol{G}, \mathcal{S}_{\text{seq}}^{\boldsymbol{G}})$ or $(\mathcal{C}^{\boldsymbol{F}}, \boldsymbol{F})$. $\mathcal{D}_{\text{seq}}$ runs $\mathcal{D}_{\text{pub}}$ (see Figure 4.3). $\mathcal{D}_{\text{seq}}$ simply relays any F-query of $\mathcal{D}_{\text{pub}}$ to its own F oracle, returning the corresponding answer. When $\mathcal{D}_{\text{pub}}$ makes a G-query x, $\mathcal{D}_{\text{seq}}$ makes all the necessary F-queries to its own F-oracle to compute $\mathcal{C}^F(x)$ and returns this value as the answer to $\mathcal{D}_{\text{pub}}$. Once $\mathcal{D}_{\text{pub}}$ has returned 0 or 1, $\mathcal{D}_{\text{seq}}$ makes all the G-queries that have been made by $\mathcal{D}_{\text{pub}}$ to its own G-oracle and checks whether all the answers it has given to $\mathcal{D}_{\text{pub}}$ (by computing $\mathcal{C}^F$ with its own F-oracle) correspond (in which case we say that event `check` happens). If this is the case, $\mathcal{D}_{\text{seq}}$ returns the same answer as $\mathcal{S}_{\text{pub}}$. Otherwise it returns 1. Note that $\mathcal{D}_{\text{seq}}$ is indeed sequential, and that its total oracle queries cost is less than $2q$ when $\mathcal{D}_{\text{pub}}$'s total oracle queries cost is less than q.

First, it is straightforward to verify that when $\mathcal{D}_{\text{seq}}$ interacts with $(\mathcal{C}^{\boldsymbol{F}}, \boldsymbol{F})$, `check` happens with probability one so that

$$\Pr\left[\mathcal{D}_{\text{seq}}^{\mathcal{C}^{\boldsymbol{F}},\boldsymbol{F}}=1\right] = \Pr\left[\mathcal{D}_{\text{pub}}^{\mathcal{C}^{\boldsymbol{F}},\boldsymbol{F}}=1\right] \quad .$$

When $\mathcal{D}_{\text{seq}}$ interacts with $(\boldsymbol{G}, \mathcal{S}_{\text{seq}}^{\boldsymbol{G}})$, one can write:

$$\Pr\left[\mathcal{D}_{\text{seq}}^{\boldsymbol{G},\mathcal{S}_{\text{seq}}^{\boldsymbol{G}}}=1\right] = \Pr\left[\mathcal{D}_{\text{seq}}^{\boldsymbol{G},\mathcal{S}_{\text{seq}}^{\boldsymbol{G}}}=1 \,|\, \texttt{check}\right] \Pr[\texttt{check}] + \Pr[\overline{\texttt{check}}] \quad .$$

Note that when event `check` occurs, all answers to F- and G-queries of $\mathcal{D}_{\text{pub}}$ have been answered as if $\mathcal{D}_{\text{pub}}$ had been interacting directly with $(\boldsymbol{G}, \mathcal{S}_{\text{pub}}^{\overline{\boldsymbol{G}}})$. This follows from the definition of $\mathcal{S}_{\text{pub}}$ and the fact that $\boldsymbol{G}$ is stateless. The statelessness of $\boldsymbol{G}$ is crucial here since even when `check` happens, the sequence of queries to $\boldsymbol{G}$ when $\mathcal{D}_{\text{seq}}$ interacts with $(\boldsymbol{G}, \mathcal{S}_{\text{seq}}^{\boldsymbol{G}})$ is not necessarily the same as when $\mathcal{D}_{\text{pub}}$ interacts with $(\boldsymbol{G}, \mathcal{S}_{\text{pub}}^{\overline{\boldsymbol{G}}})$: in the former, the G-queries of $\mathcal{D}_{\text{pub}}$ are forwarded to $\boldsymbol{G}$ by $\mathcal{D}_{\text{seq}}$ only once $\mathcal{D}_{\text{pub}}$ has returned, whereas in the later $\boldsymbol{G}$ receives $\mathcal{D}_{\text{pub}}$'s queries immediately. Hence we have:

$$\begin{aligned}
\Pr\left[\mathcal{D}_{\text{seq}}^{\boldsymbol{G},\mathcal{S}_{\text{seq}}^{\boldsymbol{G}}}=1\right] &= \Pr\left[\mathcal{D}_{\text{pub}}^{\boldsymbol{G},\mathcal{S}_{\text{pub}}^{\overline{\boldsymbol{G}}}}=1 \,|\, \texttt{check}\right] \Pr[\texttt{check}] + \Pr[\overline{\texttt{check}}] \\
&= \Pr\left[\mathcal{D}_{\text{pub}}^{\boldsymbol{G},\mathcal{S}_{\text{pub}}^{\overline{\boldsymbol{G}}}}=1\right] + \Pr[\overline{\texttt{check}}] \left(1 - \Pr\left[\mathcal{D}_{\text{pub}}^{\boldsymbol{G},\mathcal{S}_{\text{pub}}^{\overline{\boldsymbol{G}}}}=1 \,|\, \overline{\texttt{check}}\right]\right) \\
&\geq \Pr\left[\mathcal{D}_{\text{pub}}^{\boldsymbol{G},\mathcal{S}_{\text{pub}}^{\overline{\boldsymbol{G}}}}=1\right] \quad .
\end{aligned}$$

It follows from assumption (4.1) that

$$\left|\Pr\left[\mathcal{D}_{\text{pub}}^{\boldsymbol{G},\mathcal{S}_{\text{pub}}^{\overline{\boldsymbol{G}}}}=1\right] - \Pr\left[\mathcal{D}_{\text{pub}}^{\mathcal{C}^{\boldsymbol{F}},\boldsymbol{F}}=1\right]\right| \leq \left|\Pr\left[\mathcal{D}_{\text{seq}}^{\boldsymbol{G},\mathcal{S}_{\text{seq}}^{\boldsymbol{G}}}=1\right] - \Pr\left[\mathcal{D}_{\text{seq}}^{\mathcal{C}^{\boldsymbol{F}},\boldsymbol{F}}=1\right]\right| \quad ,$$

which is less than ϵ since by hypothesis $\mathcal{C}^{\boldsymbol{F}}$ is $(2q, \sigma, \epsilon)$-seq-indifferentiable from $\boldsymbol{G}$. The result follows by noting that $\mathcal{S}_{\text{pub}}$ makes at most q `Reveal` queries and σ queries to $\boldsymbol{G}$.

Clearly, $\mathcal{S}_{\text{pub}}$ and $\mathcal{D}_{\text{pub}}$ are polynomial-time if $\mathcal{S}_{\text{seq}}$ and $\mathcal{D}_{\text{seq}}$ are, so that pub-indifferentiability holds computationally if seq-indifferentiability does. □

A very simple example enables to separate full indifferentiability from seq/pub-indifferentiability, namely the Merkle-Damgård construction without strengthening using a random compression function: it was proven in [CDMP05] that it is not indifferentiable from a random oracle (a consequence of length-extension attacks), and in [DRS09] that it is pub-indifferentiable from a random oracle.

4.3.1 Separation between public and sequential indifferentiability for Stateful Ideal Primitives

When the ideal primitive $\boldsymbol{G}$ is stateful, then seq-indifferentiability does not necessarily imply pub-indifferentiability in the computational setting, as was observed by Ristenpart[5]. To see this, let $\mathcal{E} = (\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$ be a IND-CPA public-key encryption scheme. The ideal primitive $\boldsymbol{G}$ maintains a hashtable T with n-bit keys and takes as input an n-bit string x and a public key `pk` for $\mathcal{E}$.

1: **procedure** $\boldsymbol{G}(x,\texttt{pk})$
2: **if** $T(x) = \perp$ **then**
3: $y \leftarrow_{\mathcal{R}} \{0,1\}^n$
4: $T(x) := \texttt{Enc}_{\texttt{pk}}(y)$
5: **end if**
6: **return** $T(x)$
7: **end procedure**

The construction $\mathcal{C}$ is quite similar to $\boldsymbol{G}$, but instead of drawing a uniformly random y it uses a random function oracle $\boldsymbol{F} : \{0,1\}^n \to \{0,1\}^n$:

1: **procedure** $\mathcal{C}^{\boldsymbol{F}}(x,\texttt{pk})$
2: **if** $T(x) = \perp$ **then**
3: $y := \boldsymbol{F}(x)$
4: $T(x) := \texttt{Enc}_{\texttt{pk}}(y)$
5: **end if**
6: **return** $T(x)$
7: **end procedure**

One can show that $\mathcal{C}^{\boldsymbol{F}}$ is computationally seq-indifferentiable from $\boldsymbol{G}$, but not pub-indifferentiable. The idea is that in the seq-indifferentiability game, the simulator can always get the y values drawn by $\boldsymbol{G}$ by generating the public keys by itself, whereas in the pub-indifferentiability game, when the distinguisher makes a G-query before the simulator, the y value will be hidden to $\mathcal{S}_{\text{pub}}$ unless it can break the one-wayness of $\mathcal{E}$. The seq-indifferentiability simulator maintains an history F for the simulated oracle and is defined as follows:

[5]Personal communication

procedure $\mathcal{S}^{\boldsymbol{G}}_{\text{seq}}(x)$
  if $F(x) = \perp$ **then**
    $(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{KeyGen}$
    $c := \boldsymbol{G}(x, \mathtt{pk})$
    $y := \mathtt{Dec}_{\mathtt{sk}}(c)$
    $F(x) := y$
  end if
  return $F(x)$
end procedure

It is not very hard to see that the above simulator works for seq-indifferentiability. On the other hand, consider the following distinguisher for pub-indifferentiability:

procedure $\mathcal{D}^{\Theta_1,\Theta_2}_{\text{pub}}(1^n)$
  $(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{KeyGen}$
  $c := \Theta_1(0, \mathtt{pk})$
  $y := \mathtt{Dec}_{\mathtt{sk}}(c)$
  $y' := \Theta_2(0)$
  if $y = y'$ **then**
    return 1
  else
    return 0
  end if
end procedure

When $(\Theta_1, \Theta_2) = (\mathcal{C}^{\boldsymbol{F}}, \boldsymbol{F})$ then $\mathcal{D}_{\text{pub}}$ always returns 1. However any efficient simulator $\mathcal{S}_{\text{pub}}$ such that $\mathcal{D}_{\text{pub}}$ returns 1 with non negligible probability when $(\Theta_1, \Theta_2) = (\boldsymbol{G}, \mathcal{S}^{\boldsymbol{G}}_{\text{pub}})$ can be turned into an algorithm breaking the one-wayness of $\mathcal{E}$. Hence $\mathcal{D}_{\text{pub}}$ distinguishes the two systems with overwhelming probability.

4.4 Sequential Distinguisher for the 5-Round Feistel Construction

The sequential distinguisher $\mathcal{D}$ proceeds as follows (see Figure 4.4). It chooses an arbitrary value Z_{13}, two arbitrary values Y_{14} et Y_{23}, and queries $F_3(Y_{14})$ and $F_3(Y_{23})$. It then computes:

$$\begin{cases} X_{12} = Z_{13} \oplus F_3(Y_{14}) \\ X_{34} = Z_{13} \oplus F_3(Y_{23}) \end{cases}.$$

Notations are chosen such that input round values sharing a common index correspond to the same input-output pair of the Feistel scheme: we say they constitute a chain. For example, (X_{12}, Y_{14}, Z_{13}) constitute a chain since $X_{12} = Z_{13} \oplus F_3(Y_{14})$. The distinguisher then queries $F_2(X_{12})$ and $F_2(X_{34})$ and computes:

$$\begin{cases} R_1 = Y_{14} \oplus F_2(X_{12}) \\ R_2 = Y_{23} \oplus F_2(X_{12}) \\ R_3 = Y_{23} \oplus F_2(X_{34}) \\ R_4 = Y_{14} \oplus F_2(X_{34}) \end{cases}.$$

Note that necessarily $R_1 \oplus R_2 \oplus R_3 \oplus R_4 = 0$.
Then the distinguisher queries $F_1(R_1)$, $F_1(R_2)$, $F_1(R_3)$, and $F_1(R_4)$ and computes:

$$\begin{cases} L_1 = X_{12} \oplus F_1(R_1) \\ L_2 = X_{12} \oplus F_1(R_2) \\ L_3 = X_{34} \oplus F_1(R_3) \\ L_4 = X_{34} \oplus F_1(R_4) \ . \end{cases}$$

Finally the distinguisher makes the P-queries $(S_1, T_1) = P(0, (L_1, R_1))$, $(S_2, T_2) = P(0(L_2, R_2))$, $(S_3, T_3) = P(0, (L_3, R_3))$ and $(S_4, T_4) = P(0, (L_4, R_4))$. If $S_1 \oplus S_2 \oplus S_3 \oplus S_4 = 0$, it returns 1, otherwise it returns 0. Note that this distinguisher is sequential.

First, one can easily verify that $\mathcal{D}$ always returns 1 when it interacts with $(\Psi_5^{\boldsymbol{F}}, \boldsymbol{F})$. Indeed, denote $Z_{24} = X_{12} \oplus F_3(Y_{23})$ the input value to F_4 associated with (L_2, R_2). Since $X_{12} \oplus F_3(Y_{14}) = X_{34} \oplus F_3(Y_{23}) = Z_{13}$, then $Z_{24} = X_{34} \oplus F_3(Y_{14})$, so that Z_{24} is also the input value to F_4 associated with (L_4, R_4). It follows that:

$$\begin{cases} S_1 = Y_{14} \oplus F_4(Z_{13}) \\ S_2 = Y_{23} \oplus F_4(Z_{24}) \\ S_3 = Y_{23} \oplus F_4(Z_{13}) \\ S_4 = Y_{14} \oplus F_4(Z_{24}) \ , \end{cases}$$

and the relation $S_1 \oplus S_2 \oplus S_3 \oplus S_4 = 0$ is always verified.

On the contrary, when interacting with $(\boldsymbol{P}, \mathcal{S}^{\boldsymbol{P}})$, it returns 1 only with negligible probability. Indeed, considering the union of $\mathcal{D}$ and $\mathcal{S}$ as a single machine making a polynomial number of queries to the random permutation $\boldsymbol{P}$, it can find four input/output pairs $(S_i, T_i) = \boldsymbol{P}(0, (L_i, R_i))$ satisfying $R_1 \oplus R_2 \oplus R_3 \oplus R_4 = 0$ and $S_1 \oplus S_2 \oplus S_3 \oplus S_4 = 0$ only with negligible probability.

4.5 Seq-Indifferentiability of the 6-Round Feistel Construction

In this section we prove the main result of this section which states that the Feistel construction with 6 rounds and random round functions is seq-indifferentiable from a random invertible permutation, and hence also pub-indifferentiable since a random invertible permutation is stateless. Before stating the result, we recall that in [CPS08], it was shown that the Feistel construction with five rounds is not indifferentiable from a random invertible permutation. In fact, the distinguisher they described is sequential, which implies that the 5-round Feistel construction is not even seq-indifferentiable from a random invertible permutation. We recalled the attack in the previous section.

Theorem 4.2. *The Feistel construction with six rounds and random round functions is (q, t, σ, ϵ)-seq-indifferentiable from a random invertible permutation, where:*

$$\sigma(q) = q^2 \quad \textit{and} \quad \epsilon(q) = \frac{8q^4}{2^n} + \frac{q^4}{2^{2n}} \quad t(q) = \mathcal{O}(q^2 n) \ .$$

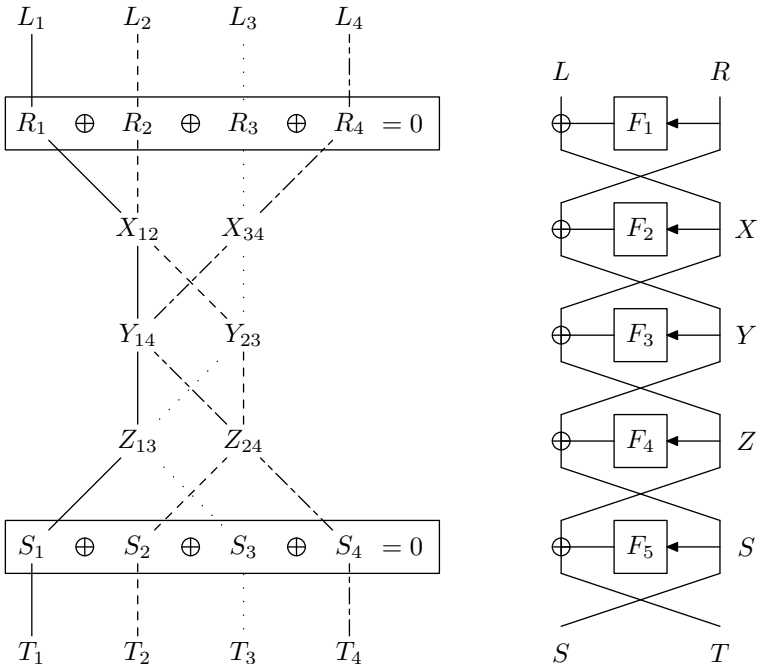

Figure 4.4: Description of the sequential distinguisher for the 5-round Feistel construction.

The rest of this section is devoted to the proof of Theorem 4.2. We will consider a sequential distinguisher $\mathcal{D}$ that first issues at most q_f queries to the simulator (or the random functions $\boldsymbol{F}_i$). These queries will be called F-queries. Then, it issues at most q_p queries to the random permutation $\boldsymbol{P}$ (or the Feistel construction $\Psi_6^{\boldsymbol{F}}$). These queries will be called P-queries. The total oracle queries cost is $q_f + 6q_p$ (for each P-query, the Feistel construction makes 6 F-queries to compute the answer) and is assumed to be less than q.

We start by describing how the simulator $\mathcal{S}$ works. It maintains an history of values for which each round function has been defined (either because this value has been queried by the distinguisher, or because the simulator has set this value internally). We will note F_i, $i \in [1..6]$ the history of the i-th round function, that is a set of pairs $(U, V) \in \{0,1\}^n \times \{0,1\}^n$, where U is an input to round function F_i and V is the corresponding image (which we denote $F_i(U) = V$). We write $U \in F_i$ to denote that the image of U by F_i is defined in the history. Initially round function values $F_i(U)$ are undefined for all $i \in [1..6]$ and all $U \in \{0,1\}^n$. The images are then modified during the execution of the simulator. $F_i(U) \leftarrow V$ means that the image of U by F_i is set to V and $F_i(U) \leftarrow_{\mathcal{R}} \{0,1\}^n$ means that the image of U by F_i is set uniformly at random in $\{0,1\}^n$. If a round function value is already in the history and a new assignment occurs, the previous value is overwritten (alternatively, we could let the simulator abort in this case, as in [CPS08], but as we will see this happens only with negligible probability so that the exact behavior of the simulator

in such a case in unessential). We will note $\mathcal{H} = (F_1, \ldots, F_6)$ the complete history of the six round functions.

When the simulator receives a F-query (i, U) (meaning that the distinguisher asks for the image of U through round function F_i), it calls an internal procedure `Query`(i, U). This procedure checks whether the corresponding image is in the history of F_i, in which case it returns this value and stops. Otherwise it sets the image uniformly at random. If $i = 1$, 2, 5, or 6, it does nothing more. If $i = 3$ or 4, the simulator additionally completes all centers $(Y, Z) \in F_3 \times F_4$ newly created so that the corresponding values of (L, R) and (S, T) obtained by evaluating the Feistel construction respectively backward and forward are consistent with the random permutation $\boldsymbol{P}$, meaning that $\boldsymbol{P}(0, (L, R)) = (S, T)$. This is done by calling two internal procedures `CompleteForward` (if $i = 4$) or `CompleteBackward` (if $i = 3$) which "adapts" two round function values ($F_5(A)$ and $F_6(S)$ for `CompleteForward`, and $F_1(R)$ and $F_2(X)$ for `CompleteBackward`) so that the Feistel matches with the random permutation. The pseudo-code for the three procedures is given below. Statements put in boxes in `CompleteForward` and `CompleteBackward` are replacements for a different system used in the indifferentiability proof and can be ignored for the moment.

There are two points to prove in order to obtain Theorem 4.2: that the simulator runs in polynomial time, and then that the probabilities that the distinguisher outputs 1 when interacting with $(\boldsymbol{P}, \mathcal{S}^{\boldsymbol{P}})$ and $(\Psi_6^{\boldsymbol{F}}, \boldsymbol{F})$ differ by a negligible quantity ϵ. The following lemma shows that the simulator runs in time polynomial in the number of queries it receives.

Lemma 4.1. *When the simulator is asked at most q queries, then the size of histories for F_3 and F_4 is at most q, the size of histories for F_1, F_2, F_5 and F_6 is at most $q^2 + q$, the procedures* `CompleteForward` *and* `CompleteBackward` *are called in total at most q^2 times, and the simulator makes at most q^2 queries to the random permutation.*

Proof. Elements are added to the history of F_3 and F_4 only when a corresponding F-query is made to the simulator, so that the size of their history cannot be greater than q. For each pair $(Y, Z) \in F_3 \times F_4$, either `CompleteForward`$(Y, Z)$ or `CompleteBackward`(Y, Z) is called, at most once, so that in total these procedures are called at most q^2 times. Since the simulator makes one query to the random permutation per execution of `CompleteForward` and `CompleteBackward` this in turns implies that the total number of queries to $\boldsymbol{P}$ is at most q^2. Finally, elements are added to the history of F_1, F_2, F_5 and F_6 either when a query is made to the simulator, or during an execution of `CompleteForward` or `CompleteBackward`, so that the size of their history cannot be greater than $q^2 + q$. □

In order to prove that the two systems $\Sigma_1 = (\boldsymbol{P}, \mathcal{S}^{\boldsymbol{P}})$ and $\Sigma_4 = (\Psi_6^{\boldsymbol{F}}, \boldsymbol{F})$ are indistinguishable, we will use two intermediate systems: $\Sigma_2 = (\Psi_6^{\boldsymbol{S}^{\boldsymbol{P}}}, \boldsymbol{S}^{\boldsymbol{P}})$ where the P-queries of $\mathcal{D}$ are answered by the Feistel construction asking round function values to the simulator, which itself interacts with $\boldsymbol{P}$, and $\Sigma_3 = (\Psi_6^{\boldsymbol{S}^{\boldsymbol{R}}}, \boldsymbol{S}^{\boldsymbol{R}})$ where the random invertible permutation is replaced by a two-sided random function $\boldsymbol{R}$ (note

Algorithm 1 Simulator

variable: round function histories $F_1, \ldots, F_6$

procedure `Query`(i,U)
  if $U \notin F_i$ **then**
    $F_i(U) \leftarrow_{\mathcal{R}} \{0,1\}^n$
    if $i = 3$ **then**
      for all $Z \in F_4$ **do**
        `CompleteBackward`(U, Z)
      end for
    end if
    if $i = 4$ **then**
      for all $Y \in F_3$ **do**
        `CompleteForward`(Y, U)
      end for
    end if
  end if
  return $F_i(U)$
end procedure

procedure `CompleteForward`(Y,Z)
  $X := Z \oplus F_3(Y)$
  `Query`$(2, X)$
  $R := Y \oplus F_2(X)$
  `Query`$(1, R)$
  $L := X \oplus F_1(R)$
  $(S,T) := \boldsymbol{P}(0,(L,R))$ $\quad$ $\boxed{(S,T) := \boldsymbol{R}(0,(L,R))}$
  $A := Y \oplus F_4(Z)$
  $F_5(A) \leftarrow Z \oplus S$
  $F_6(S) \leftarrow A \oplus T$
end procedure

procedure `CompleteBackward`(Y,Z)
  $A := Y \oplus F_4(Z)$
  `Query`$(5, A)$
  $S := Z \oplus F_5(A)$
  `Query`$(6, S)$
  $T := A \oplus F_6(S)$
  $(L,R) := \boldsymbol{P}(1,(S,T))$ $\quad$ $\boxed{(L,R) := \boldsymbol{R}(1,(S,T))}$
  $X := Z \oplus F_3(Y)$
  $F_2(X) \leftarrow R \oplus Y$
  $F_1(R) \leftarrow L \oplus X$
end procedure

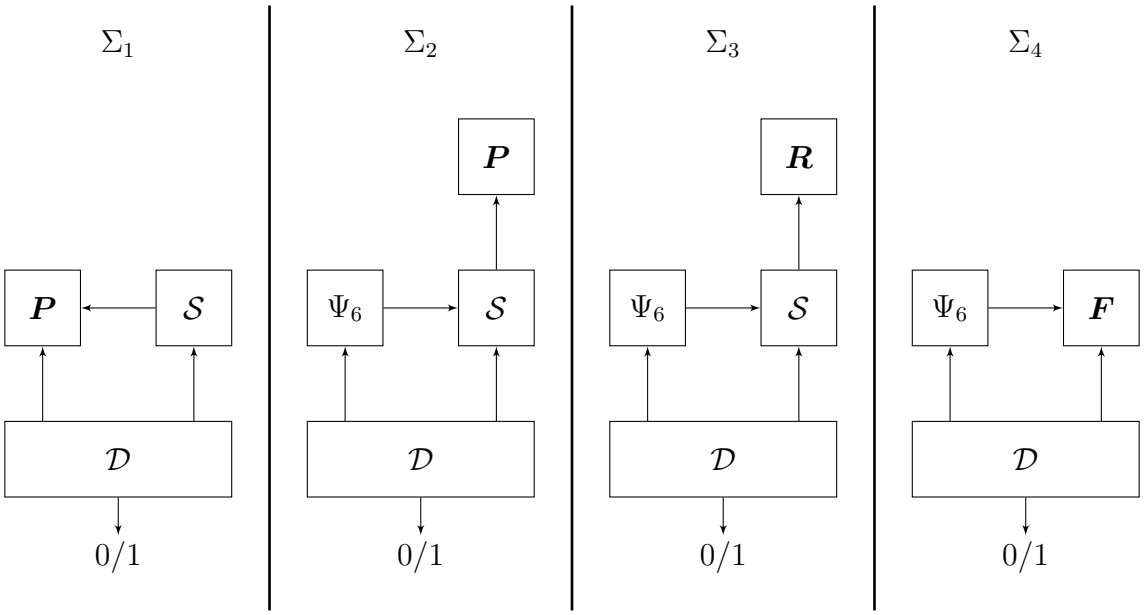

Figure 4.5: Systems used in the seq-indifferentiability proof.

the corresponding change in procedures `CompleteForward` and `CompleteBackward` indicated by a boxed statement). The four systems used in the proof are depicted in Figure 4.5.

The main part of the analysis is concerned with systems Σ_2 and Σ_3. We will show that unless some bad event happens, the round function values set by the simulator in Σ_2 are consistent with $\boldsymbol{P}$ (which will enable to bound the statistical distance between Σ_1 and Σ_2), and that in Σ_3 they are uniformly random and independent (which will enable to bound the statistical distance between Σ_3 and Σ_4). In systems Σ_2 and Σ_3, the simulator first receives at most q_f queries from the distinguisher, and then at most $6q_p$ queries from the Feistel construction (6 for each P-query of the distinguisher). Hence the total number of queries received by the simulator is exactly the total oracle queries cost of $\mathcal{D}$, which is less than q. The statistical distance between answers of systems Σ_2 and Σ_3 is easily bounded.

Lemma 4.2. *For any distinguisher of total oracle queries cost at most q, the following holds:*

$$\left|\Pr\left[\mathcal{D}^{\Sigma_2}=1\right]-\Pr\left[\mathcal{D}^{\Sigma_3}=1\right]\right| \leq \frac{q^4}{2^{2n+1}} \ .$$

Proof. Consider the union of $\mathcal{D}$, Ψ_6, and $\mathcal{S}$ as a single distinguisher $\mathcal{D}'$ interacting either with a random invertible permutation or a two-sided random function. Note that $\mathcal{D}'$ makes at most q^2 queries to its oracle (Lemma 4.1). One can conclude thanks to the PRF/PRP switching lemma [BR06]. □

Before going further with the proof, we define formally what it means for an

input $x \in \{0,1\} \times \{0,1\}^n$ to the Feistel construction to be computable with respect to the history of the simulator.

Definition 4.4 (Computable input)**.** *Given a simulator history $\mathcal{H}$ and an input $x \in \{0,1\} \times \{0,1\}^{2n}$, the sequence $\rho_{\mathcal{H}}(x) = (\rho_{\mathcal{H}}(x)[i])_{i\in[0..7]}$ is defined as follows:*

- *for a forward input $x = (0,(L,R))$, $\rho_{\mathcal{H}}(x)[0] = L$, $\rho_{\mathcal{H}}(x)[1] = R$, and for $i = 2$ to 7:*

$$\begin{cases} \text{if } \rho_{\mathcal{H}}(x)[i-1] \in F_{i-1} \text{ then } \rho_{\mathcal{H}}(x)[i] = \rho_{\mathcal{H}}(x)[i-2] \oplus F_{i-1}(\rho_{\mathcal{H}}(x)[i-1]) \\ \text{else } \rho_{\mathcal{H}}(x)[i] = \perp \end{cases}$$

- *for a backward input $x = (1,(S,T))$, $\rho_{\mathcal{H}}(x)[7] = T$, $\rho_{\mathcal{H}}(x)[6] = S$, and for $i = 5$ to 0:*

$$\begin{cases} \text{if } \rho_{\mathcal{H}}(x)[i+1] \in F_{i+1} \text{ then } \rho_{\mathcal{H}}(x)[i] = \rho_{\mathcal{H}}(x)[i+2] \oplus F_{i+1}(\rho_{\mathcal{H}}(x)[i+1]) \\ \text{else } \rho_{\mathcal{H}}(x)[i] = \perp \end{cases}$$

An input x is said to be computable with respect to $\mathcal{H}$ *iff $\rho_{\mathcal{H}}(x)[i] \neq \perp$ for all $i \in [0..7]$. In that case we note $\Psi_6^{\mathcal{H}}(x) = (\rho_{\mathcal{H}}(x)[6], \rho_{\mathcal{H}}(x)[7])$ if x is a forward input and $\Psi_6^{\mathcal{H}}(x) = (\rho_{\mathcal{H}}(x)[0], \rho_{\mathcal{H}}(x)[1])$ if x is a backward input.*

For a computable input x, we will often use the notation $(L, R, X, Y, Z, A, S, T) = \rho_{\mathcal{H}}(x)$ as depicted on Figure 4.1.

We now define a bad event that may occur during the execution of the simulator (in Σ_2 or Σ_3) in relation with Lines 26, 27, 37, and 38 of the simulator. We will say that event `Bad` happens if in any execution of `CompleteForward` or `CompleteBackward`, the input value whose image is set at Lines 26, 27, 37 or 38 is already in the history of the corresponding round function. This implies that the simulator overwrites a value so that its answers may not be coherent with $\boldsymbol{P}$ or $\boldsymbol{R}$ any more.[6] Reciprocally, if `Bad` does not happen, then the simulator never overwrites any value in its history.

We start with the simple observation that if `Bad` does not happen, then during any execution of `CompleteForward` or `CompleteBackward`, the query to $\boldsymbol{P}$ or $\boldsymbol{R}$ made by the simulator is fresh.

Lemma 4.3. *In system Σ_2, if* `Bad` *does not happen, then in any execution of* `CompleteForward` *or* `CompleteBackward` *the query to $\boldsymbol{P}$ made by the simulator is not in the history of $\boldsymbol{P}$. For system Σ_3, the corresponding statement holds for $\boldsymbol{R}$.*

Proof. The reasoning is the same for Σ_2 and Σ_3, we use Σ_2 to fix ideas. Consider an execution of `CompleteForward`(Y,Z). Let $x = (0,(L,R))$ be the query to $\boldsymbol{P}$ made by the simulator, and $(S,T) = \boldsymbol{P}(x)$. If x is already in the history of $\boldsymbol{P}$,

[6] In previous work on indifferentiability of the Feistel construction [CPS08, Seu09], in such a case the simulator aborted. It does not change much since, as we will prove, this happens only with negligible probability.

then it was necessarily added by a previous execution of $\texttt{CompleteForward}(Y', Z')$ or $\texttt{CompleteBackward}(Y', Z')$ (note that the distinguisher does not make any query to $\boldsymbol{P}$ in Σ_2 or to $\boldsymbol{R}$ in Σ_3). But since Bad does not happen, round function values are never overwritten so that necessarily $(Y', Z') = (Y, Z)$. This is impossible since by construction the simulator makes at most one call to CompleteForward or CompleteBackward per center $(Y, Z) \in F_3 \times F_4$. □

We are now ready to upper bound the probability that Bad happens in Σ_2 or Σ_3.

Lemma 4.4. *For any distinguisher of total oracle queries cost at most q, event* Bad *happens with probability less than $4q^4/2^n$ in Σ_3 and less than $4q^4/2^n + q^4/2^{2n+1}$ in Σ_2.*

Proof. We start by working with Σ_3 since it is slightly easier to analyze. Assume Bad has not happened yet, and consider a call to $\texttt{Query}(3, Y)$ (the case of $\texttt{Query}(4, Z)$ is symmetric). Let $Z_1, \ldots, Z_m$, ($m \leq q$ according to Lemma 4.1) be the values in the history of F_4 at this point. We show that event Bad does not happen for any call to $\texttt{CompleteBackward}(Y, Z_i)$ except with negligible probability. Since $F_3(Y)$ is set uniformly at random, the probability that any value $X_i = Z_i \oplus F_3(X)$ is in the history of F_2 at the time $F_3(Y)$ is set is less than $m(q^2 + q)/2^n \leq 2q^3/2^n$. Moreover $F_2(X_i)$ cannot be set until $\texttt{CompleteBackward}(Y, Z_i)$ is called, hence Bad does not happen for Line 37 of any execution of $\texttt{CompleteBackward}(Y, Z_i)$ except with probability less than $2q^3/2^n$. Let (L_i, R_i) denote the answer of the query to $\boldsymbol{R}$ in $\texttt{CompleteBackward}(Y, Z_i)$. Since Bad has not happened yet, according to Lemma 4.3 this query is not in the history of $\boldsymbol{R}$ so that the answer is uniformly random. Hence R_i is in the history of F_1 with probability less than $(q^2 + q)/2^n$. Since there are $m \leq q$ calls to CompleteBackward, event Bad does not occur for Line 38 of CompleteBackward except with probability less than $2q^3/2^n$. Finally, since there are at most q calls in total to $\texttt{Query}(3, \cdot)$ and $\texttt{Query}(4, \cdot)$, event Bad happens with probability less than $4q^4/2^n$ in Σ_3. The absolute difference between the probability that Bad happens in Σ_2 and Σ_3 cannot be greater than the statistical distance between answers of $\boldsymbol{P}$ and $\boldsymbol{R}$, hence the probability that Bad happens in Σ_2 is less than $4q^4/2^n + q^4/2^{2n+1}$. □

The following lemma says that as long as Bad does not happen in Σ_2, the round function values set by the simulator are consistent with $\boldsymbol{P}$.

Lemma 4.5. *If* Bad *does not happen in system Σ_2, then for any input $x \in \{0,1\} \times \{0,1\}^{2n}$ computable with respect to the final history of the simulator $\mathcal{H}$, $\Psi_6^{\mathcal{H}}(x) = \boldsymbol{P}(x)$.*

Proof. Consider an input $x \in \{0,1\} \times \{0,1\}^{2n}$ computable with respect to the final history $\mathcal{H}$ of the simulator, and let $(L, R, X, Y, Z, A, S, T) = \rho_{\mathcal{H}}(x)$. There was necessarily a call to $\texttt{CompleteForward}(Y, Z)$ or $\texttt{CompleteBackward}(Y, Z)$ during the execution of the simulator. With respect to the history $\mathcal{H}'$ just after the completion of $\texttt{CompleteForward}(Y, Z)$ or $\texttt{CompleteBackward}(Y, Z)$, it is clear that $\Psi_6^{\mathcal{H}'}(x) =$

$\boldsymbol{P}(x)$. Since Bad does not happen the simulator never overwrites a value and the equality remains true until the end of the simulation, hence $\Psi_6^{\mathcal{H}}(x) = \boldsymbol{P}(x)$. □

A direct consequence of this lemma is that as long as Bad does not happen in Σ_2, the answers of systems Σ_1 and Σ_2 are identically distributed.

Lemma 4.6. *For any distinguisher of total oracle queries cost at most q, the following holds:*

$$\left|\Pr\left[\mathcal{D}^{\Sigma_1} = 1\right] - \Pr\left[\mathcal{D}^{\Sigma_2} = 1\right]\right| \leq \frac{4q^4}{2^n} + \frac{q^4}{2^{2n+1}} \ .$$

Proof. Clearly, answers to F-queries of the distinguisher are identically distributed in Σ_1 and Σ_2 since they are answered by $\mathcal{S}^{\boldsymbol{P}}$ in both systems (may Bad occur or not).[7] Moreover, in Σ_2 any P-query x asked by the distinguisher is computable with respect to the history of the simulator at the time it is answered by Ψ_6, and if Bad does not happen in Σ_2, then according to Lemma 4.5, $\Psi_6^{\mathcal{H}}(x) = \boldsymbol{P}(x)$ so that answers to P-queries of the distinguisher are also identically distributed in both systems. The result follows from Lemma 4.4. □

Lemma 4.7. *If* Bad *does not happen in system Σ_3, then the round function values set by the simulator are uniformly random and independent.*

Proof. Since this is clear for round function values set uniformly at random (independently of Bad occurring or not), we only have to examine values that are adapted at Lines 26, 27, 37, and 38 of the simulator. But according to Lemma 4.3, if Bad does not happen, the query to $\boldsymbol{R}$ made by the distinguisher in any execution of CompleteForward or CompleteBackward is not in the history of $\boldsymbol{R}$, so that the answer (S, T) or (L, R) is uniformly random. Consequently, round function values set by $F_5(A) \leftarrow Z \oplus S$ and $F_6(S) \leftarrow A \oplus T$ in CompleteForward, or $F_2(X) \leftarrow R \oplus Y$ and $F_1(R) \leftarrow L \oplus X$ in CompleteBackward are uniformly random and independent of previous round function values set by the simulator. Since Bad does not happen round function values are not overwritten and the result follows. □

This lemma finally enables to bound the statistical distance between the answers of Σ_3 and Σ_4.

Lemma 4.8. *For any distinguisher of total oracle queries cost at most q, the following holds:*

$$\left|\Pr\left[\mathcal{D}^{\Sigma_3} = 1\right] - \Pr\left[\mathcal{D}^{\Sigma_4} = 1\right]\right| \leq \frac{4q^4}{2^n} \ .$$

Proof. If Bad does not occur in Σ_3 then answers of $\mathcal{S}^{\boldsymbol{R}}$ are distributed exactly as answers of $\boldsymbol{F}$ according to Lemma 4.7. Hence the statistical distance between answers of Σ_3 and Σ_4 is upper bounded by the probability that Bad happens in Σ_3, given by Lemma 4.4. □

[7] It is crucial here that the distinguisher is sequential, otherwise the simulation in Σ_2 would be altered by the queries made by Ψ_6.

Theorem 4.2 is now a simple consequence of Lemmata 4.2, 4.6, and 4.8.

Remark 4.1 The strategy of using the intermediate system Σ_2 is likely to be quite generic for seq-indifferentiability proofs (system Σ_3, on the contrary, is quite specific to the Feistel construction). We believe this could probably make proofs of pub-indifferentiability (*e.g.* [DRS09, Section 7]) much easier, but leave this for future work.

Remark 4.2 Note that for general distinguishers (not necessarily sequential), the proof would go through exactly as above for Lemmata 4.2 and 4.8. The problematic step is clearly going from Σ_1 to Σ_2. To see what could go wrong if the distinguisher can interleave queries to $\boldsymbol{P}$ and $\mathcal{S}$, consider the following simple example. $\mathcal{D}$ first makes a P-query $\boldsymbol{P}(0,(L,R)) = (S,T)$, and then makes the sequence of F-queries $F_1(R)$, $F_2(X)$, $F_6(S)$, $F_5(A)$. In system Σ_1, the simulator returns uniformly answers to the four F-queries and will be unable to adapt F_3 and F_4, whereas in Σ_2 the initial P-query of the distinguisher will trigger six F-queries from Ψ_6 which will lead the simulator to adapt the chain when query $F_4(Y)$ occurs. Making progress towards proving full indifferentiability for six rounds clearly requires to find the right way to deal with these "external" chains without knowing the P-queries of the distinguisher.

4.6 Applications to Correlation Intractability

Correlation intractability was introduced by Canetti *et al.* in their work on the limits of the random oracle methodology [CGH98]. In the standard model, a function family is said to be correlation intractable if given the description of a random function f of the family, no Probabilistic Polynomial Time (PPT) algorithm can find an input x, or more generally a sequence of inputs $(x_1,\ldots,x_m)$, such that $((x_1,\ldots,x_m),(f(x_1),\ldots,f(x_m)))$ satisfies a relation that would be hard to satisfy for a uniformly random function.

There is no difficulty in extending the definition of correlation intractability to an idealized model: instead of passing the description of the function as input to the algorithm, it is granted access to the ideal primitive used by the construction $\mathcal{C}$. This way one can define a correlation intractable construction (accessing an ideal primitive).

In all the following, we will consider relations over pairs of binary sequences (formally, a subset of $\{0,1\}^* \times \{0,1\}^*$). We assume that the machine $\mathcal{M}$ returns sequences of strings in $\mathtt{Dom}_n$, the domain of the ideal primitive $\boldsymbol{G}_n$ or the construction $\mathcal{C}^{\boldsymbol{F}_n}$.

Definition 4.5 (Evasive relation)**.** *Let $\boldsymbol{G} = (\boldsymbol{G}_n)$ be an ideal primitive associated to $\mathbb{G} = (\mathtt{Dom}_n, \mathtt{Rng}_n, \mathbb{G}_n)$. A relation $\mathcal{R}$ over pairs of binary sequences is said to be evasive with respect to $\boldsymbol{G}$ if for any PPT oracle machine $\mathcal{M}$, there is a negligible function ϵ such that the following holds:*

$$\Pr\left[(x_1,\ldots,x_m) \leftarrow \mathcal{M}^{\boldsymbol{G}_n} \,:\, ((x_1,\ldots,x_m),(\boldsymbol{G}_n(x_1),\ldots,\boldsymbol{G}_n(x_m))) \in \mathcal{R}\right] \leq \epsilon(n) \ .$$

Example 4.1 The relation over pairs of quadruplets of binary strings

$$\cup_n \{(((0,(L_1,R_1)),(0,(L_2,R_2)),(0,(L_3,R_3)),(0,(L_4,R_4))),((S_1,T_1),(S_2,T_2),(S_3,T_3),(S_4,T_4))) : L_i,R_i,S_i,T_i \in \{0,1\}^n \text{ and } R_1 \oplus R_2 \oplus R_3 \oplus R_4 = 0 \text{ and } S_1 \oplus S_2 \oplus S_3 \oplus S_4 = 0\}$$

is evasive for a random invertible permutation. This is exactly the evasive relation used in Section 4.4 to show that the 5-round Feistel construction is not seq-indifferentiable from a random permutation. This same attack also shows that the 5-round Feistel construction is not correlation intractable.

Definition 4.6 (Correlation intractable construction)**.** *Let $\mathcal{C}$ be a construction with oracle access to an ideal primitive $\boldsymbol{F} = (\boldsymbol{F}_n)$ and implementing some primitive $\mathbb{G}$. $\mathcal{C}^{\boldsymbol{F}}$ is said to be (multiple-output) correlation intractable if for any relation $\mathcal{R}$ over pairs of binary sequences evasive with respect to $\boldsymbol{G}$, and any PPT oracle machine $\mathcal{M}$, there is a negligible function ϵ such that:*

$$\Pr\left[(x_1,\ldots,x_m) \leftarrow \mathcal{M}^{\boldsymbol{F}_n} \,:\, \Big((x_1,\ldots,x_m),(\mathcal{C}^{\boldsymbol{F}_n}(x_1),\ldots,\mathcal{C}^{\boldsymbol{F}_n}(x_m))\Big) \in \mathcal{R}\right] \leq \epsilon(n) \ .$$

Weak correlation intractability is defined similarly as above by quantifying only over all polynomial-time recognizable relations (*i.e.* relations $\mathcal{R}$ such that there exists a polynomial-time algorithm that, given $((x_1,\ldots,x_m),(y_1,\ldots,y_m))$, decides whether it belongs to $\mathcal{R}$ or not).

Theorem 4.3. *Let $\mathcal{C}$ be a construction with oracle access to an ideal primitive $\boldsymbol{F} = (\boldsymbol{F}_n)$ and implementing some primitive $\mathbb{G}$. If $\mathcal{C}^{\boldsymbol{F}}$ is seq-indifferentiable from the ideal primitive $\boldsymbol{G}$, then $\mathcal{C}^{\boldsymbol{F}}$ is correlation intractable (resp. weakly correlation intractable).*

Proof. Assume that $\mathcal{C}^{\boldsymbol{F}}$ is not correlation intractable. Then there is an evasive relation $\mathcal{R}$ and a PPT oracle machine $\mathcal{M}$ such that $\mathcal{M}^{\boldsymbol{F}_n}$ outputs with non-negligible probability δ a sequence $(x_1,\ldots,x_m)$ such that $((x_1,\ldots,x_m),(\mathcal{C}^{\boldsymbol{F}_n}(x_1),\ldots,\mathcal{C}^{\boldsymbol{F}_n}(x_m))) \in \mathcal{R}$. Consider the following sequential distinguisher $\mathcal{D}$ accessing a pair of oracles (G,F): it runs $\mathcal{M}$, answering $\mathcal{M}$'s oracle queries with its own oracle F. $\mathcal{M}$ returns $(x_1,\ldots,x_m)$. $\mathcal{D}$ then makes oracle queries $G(x_1),\ldots,G(x_m)$ and checks[8] whether $((x_1,\ldots,x_m),(G(x_1),\ldots,G(x_m))) \in \mathcal{R}$. If this is the case it returns 1, otherwise it returns 0.

When the distinguisher is interacting with $(\mathcal{C}^{\boldsymbol{F}},\boldsymbol{F})$, the probability that it returns 1 is exactly δ, which is non-negligible by hypothesis. On the contrary, when it interacts with $(\boldsymbol{G},\mathcal{S}^{\boldsymbol{G}})$, then the union of $\mathcal{M}$ and $\mathcal{S}$ is a PPT oracle machine with oracle access to $\boldsymbol{G}$, so that by definition of an evasive relation $\mathcal{D}$ outputs 1 only with negligible probability. The advantage of the distinguisher is non-negligible, which contradicts the seq-indifferentiability of $\mathcal{C}^{\boldsymbol{F}}$. □

A direct consequence of Theorems 4.2 and 4.3 is that the 6-round Feistel construction with random round functions is correlation intractable: no polynomial

[8] Note that the reasoning holds only relatively to a polynomial-time recognizable relation if $\mathcal{D}$ is computationally bounded.

algorithm with oracle access to the round functions can find a sequence of inputs that together with their image by the Feistel satisfy a relation that would be hard to satisfy in the random invertible permutation model. Note that the sole *existence* of correlation intractable invertible permutations in the random oracle model was already implied by the result of Holenstein *et al.* [HKT11] on the full indifferentiability of the 14-round Feistel construction (since full indifferentiability implies seq-indifferentiability and hence correlation intractability), but our results shows that six rounds are sufficient to achieve this property.

Remark 4.3 According to Theorem 4.3, sequential indifferentiability implies correlation intractability. However correlation intractability does not necessarily imply sequential indifferentiability. In the following Section 4.7 we provide a simple counter-example separating the two notions.

Implications for Chosen-Key and Known-Key Attacks on Block Ciphers. Knudsen and Rijmen [KR07] have introduced so-called known-key attacks on block ciphers. We discuss the implications of our results regarding this attack model in Section 4.8.

4.7 Separating Correlation Intractability and Sequential Indifferentiability

According to Theorem 4.3, sequential indifferentiability implies correlation intractability. However, it does not hold the other way around. Below we give a constructive counter-example.

Let $\mathcal{C}^{\boldsymbol{E}}$ be a construction based on some ideal primitive $\boldsymbol{E}$ which is seq-indifferentiable from a random function $\boldsymbol{F} = (\boldsymbol{F}_n)$, $\boldsymbol{F}_n : \{0,1\}^n \to \{0,1\}^n$. By Theorem 4.3, $\mathcal{C}^{\boldsymbol{E}}$ is also correlation intractable with respect to the random function $\boldsymbol{F}$.

Now consider the primitive $\boldsymbol{G} = (\boldsymbol{G}_n)$ where $\boldsymbol{G}_n : \{0,1\}^n \to \{0,1\}^n$ is such that $\boldsymbol{G}_n(0^n) = 0^n$ and for $x \in \{0,1\}^n \setminus \{0^n\}$, we have $\boldsymbol{G}_n(x) = \boldsymbol{F}_n(x)$. Let $\mathcal{R}$ be be a relation evasive with respect to the ideal primitive $\boldsymbol{G}$. Clearly, $\mathcal{R}$ is also evasive with respect to the random function $\boldsymbol{F}$ so that $\mathcal{C}^{\boldsymbol{E}}$ is also correlation intractable with respect to $\boldsymbol{G}$. However, $\mathcal{C}^{\boldsymbol{E}}$ is not seq-indifferentiable from $\boldsymbol{G}$. Indeed, consider a distinguisher which simply makes the query 0^n to its left oracle: the answer will be 0^n when it is $\boldsymbol{G}$, and will be 0^n with only negligible probability when it is $\mathcal{C}^{\boldsymbol{E}}$ (since otherwise this would yield a sequential distinguisher distinguishing $\mathcal{C}^{\boldsymbol{E}}$ from $\boldsymbol{F}$).

4.8 Implications for Chosen-Key and Known-Key Attacks on Block Ciphers

Knudsen and Rijmen [KR07] have introduced the model of known-key attacks on block ciphers, where the attacker is given a random or chosen key K to the block cipher, and must find inputs to the block cipher that together with their image satisfy a relation that would be hard to satisfy for a random invertible permutation.

In other words, the attacker must break the correlation intractability of the block cipher for that particular key.

In the random oracle model, there are at least two straightforward ways to obtain a block cipher with a Feistel construction. The first one is to let round functions have input length $\ell + n$, where ℓ is the key length, and to prepend the key K to the input of each round function. Another way is to xor keys $(k_1, \ldots, k_r)$ (where $|k_i| = n$) to the input of the r round functions $(\boldsymbol{F}_1, \ldots, \boldsymbol{F}_r)$ (the pseudorandomness of this construction in the random oracle model has for example been studied by [GR04]). Many variations can be explored, *e.g.* having a single round function $\boldsymbol{F}$ instead of independent ones, having $\boldsymbol{F}_i$'s be random invertible permutations rather that random functions, etc.

An interesting result of [KR07] is that for a 7-round Feistel construction using a single random invertible permutation $\boldsymbol{P}$ as round function, and independent keys $(k_1, \ldots, k_7)$ xored to the input of $\boldsymbol{P}$ at each round, then the resulting block cipher is not correlation intractable (even when the keys are only random and known from the attacker, not chosen): namely with high probability on the choice of the keys, the attacker can find inputs (L, R) and (L', R') that together with the corresponding outputs (S, T) and (S', T') satisfy $R \oplus R' \oplus T \oplus T' = 0$.

Our results on the correlation intractability of the 6-round Feistel construction shows that the block cipher obtained by prepending the key to the input of each round function is correlation intractable (in the random oracle model), and hence immune to known-key and even chosen-key attacks. Other variations need more careful analysis. In particular, note that the variant using a single round function is clearly not immune to known or chosen key attacks (at least when the same key is used at each round): for example for any number of rounds, $\Psi_r^{\boldsymbol{F},\ldots,\boldsymbol{F}}(L, R) = (S, T)$ implies $\Psi_r^{\boldsymbol{F},\ldots,\boldsymbol{F}}(T, S) = (R, L)$ (this is in fact true for any palindromic sequence of round functions).

4.9 Seq-Indifferentiability Beyond the Birthday Barrier for the Construction of Chapter 3

In Chapter 3, we considered the problem of ideal cipher domain extension. We showed that a 3-round Feistel-like construction based on an n-bit ideal cipher is indifferentiable from a $2n$-bit ideal cipher. We obtained a birthday security bound, namely the construction is secure as long as the attacker makes $q \ll 2^{n/2}$ many queries. However, we also showed that the construction is actually secure up to $q \ll 2^n$ many queries in the standard indistinguishability model, where the attacker cannot make queries to the smaller n-bit ideal cipher. It was left as an open problem whether obtaining a similar improved security bound in the indifferentiability model was possible. Here we give a partial positive answer to that question, namely showing that the 3-round Feistel-like construction is seq-indifferentiable and pub-indifferentiable from an ideal cipher up to $q \ll 2^n$ queries.

The 3-round permutation $\Psi_3 : \{0,1\}^{2n} \to \{0,1\}^{2n}$ defined as follows (see Figure 4.6 for an illustration), given block ciphers E_1, E_2 and E_3 with n-bit key (first

variable) and n-bit input/output (second variable):

$$\begin{aligned} X &= E_1(R, L) \\ S &= E_2(X, R) \\ T &= E_3(S, X) \\ \Psi_3(L, R) &:= (S, T) \end{aligned}$$

The 3 round block cipher $\Psi'_3 : \{0,1\}^k \times \{0,1\}^{2n} \to \{0,1\}^{2n}$ is defined as follows, given block ciphers E_1, E_2 and E_3 with $(k+n)$-bit key and n-bit input/output:

$$\begin{aligned} X &= E_1(K\|R, L) \\ S &= E_2(K\|X, R) \\ T &= E_3(K\|S, X) \\ \Psi'_3(K, (L, R)) &:= (S, T) \end{aligned}$$

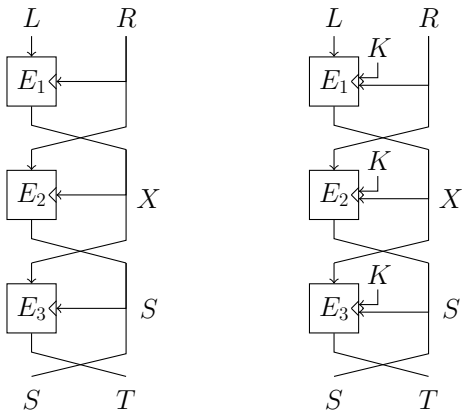

Figure 4.6: 3-round permutation $\Psi_3(L, R)$ (left) and 3-round block-cipher $\Psi'_3(K, (L, R))$ (right)

We now state our main result in this section: the 3-round Feistel construction $(\{0,1\}^{2n} \to \{0,1\}^{2n})$ is seq-indifferentiable from a random permutation up to $q \ll 2^n$ queries. To get an ideal cipher, it suffices to prepend a key K to the 3 ideal ciphers E_1, E_2 and E_3; one then gets a family of independent random permutation, parametrized by K, *i.e.* an ideal cipher.

Theorem 4.4. *The 3-round Feistel construction Ψ_3 is $(q, t, \sigma, \varepsilon)$-seq-indifferentiable from a random invertible permutation $P : \{0,1\}^{2n} \to \{0,1\}^{2n}$, with $\sigma(q) = nq$, and $\varepsilon = \mathcal{O}(q/2^n)$. The running time of the simulator for is $t = \mathcal{O}(nq \log q)$.*

We only consider the 3-round permutation Ψ_3. The extension to block-cipher Ψ'_3 is straightforward. We must construct a simulator $\mathcal{S}$ such that the two systems formed by (Ψ_3, E) and $(P, \mathcal{S})$ are indistinguishable.

Our simulator maintains a history of already answered queries for E_1, E_2 and E_3. Formally, $(1, R, L, X)$ exists in history if and only if the simulator has answered $E_1(R, L)$ query as X or $E_1^{-1}(R, X)$ query as L previously. Similar conditions hold for $(2, X, R, S)$ and $(3, S, X, T)$ as well. We define the following algorithms:

- $\mathsf{GuessE}_1(R, L)$ returns a random $X \in \{0,1\}^n \setminus \mathcal{B}$ where $\mathcal{B}$ is the set of already defined values for $E_1(R, \cdot)$. Algorithms GuessE_1^{-1}, GuessE_3 and GuessE_3^{-1} work in a similar fashion.

- $\mathsf{ConflictE}_1(R, L, X)$ returns True iff $(1, R, L, *)$ or $(1, R, *, X)$ exist in history. In other words, it checks whether $E_1(R, L)$ or $E_1^{-1}(R, X)$ has been defined before. $\mathsf{ConflictE}_2$ and $\mathsf{ConflictE}_3$ work in a similar fashion.

- $\mathsf{Store}(L, R, X, S, T)$ saves $(1, R, L, X)$, $(2, X, R, S)$ and $(3, S, X, T)$ in history.

The distinguisher's queries are answered as follows by the simulator:

$E_1(R, L)$ query:
1. If $(1, R, L, *) \in$ history
2. Return $*$
3. $X \leftarrow \mathsf{GuessE}_1(R, L)$
4. $(\mathsf{Err}, S, T) \leftarrow \mathsf{Check}(L, R, X)$
5. If $\mathsf{Err} =$ True Goto 3
6. $\mathsf{Store}(L, R, X, S, T)$
7. Return X

$E_1^{-1}(R, X)$ query:
1. If $(1, R, *, X) \in$ history
2. Return $*$
3. $L \leftarrow \mathsf{GuessE}_1^{-1}(R, X)$
4. $(\mathsf{Err}, S, T) \leftarrow \mathsf{Check}(L, R, X)$
5. If $\mathsf{Err} =$ True Goto 3
6. $\mathsf{Store}(L, R, X, S, T)$
7. Return L

$E_2(X, R)$ query:
1. If $(2, X, R, *) \in$ history
2. Return $*$
3. $L \leftarrow \mathsf{GuessE}_1^{-1}(R, X)$
4. $(\mathsf{Err}, S, T) \leftarrow \mathsf{Check}(L, R, X)$
5. If $\mathsf{Err} =$ True Goto 3
6. $\mathsf{Store}(L, R, X, S, T)$
7. Return S

$\mathsf{Check}(L, R, X)$:
1. $S \| T \leftarrow P(L \| R)$
2. If $\mathsf{ConflictE}_2(X, R, S) =$ True Or $\mathsf{ConflictE}_3(S, X, T) =$ True
3. Return (True, $*, *$)
4. Return (False, S, T).

The procedure for answering the other queries is essentially symmetric; we provide it for completeness:

$E_3^{-1}(S,T)$ query:
1. If $(3,S,*,T) \in$ history
2. Return $*$
3. $X \leftarrow \mathsf{GuessE}_3^{-1}(S,T)$
4. $(\mathsf{Err},L,R) \leftarrow \mathsf{Check}^{-1}(S,T,X)$
5. If $\mathsf{Err} =$ True Goto 3
6. $\mathsf{Store}(L,R,X,S,T)$
7. Return X

$E_3(S,X)$ query:
1. If $(3,S,X,*) \in$ history
2. Return $*$
3. $T \leftarrow \mathsf{GuessE}_3(S,X)$
4. $(\mathsf{Err},L,R) \leftarrow \mathsf{Check}^{-1}(S,T,X)$
5. If $\mathsf{Err} =$ True Goto 3
6. $\mathsf{Store}(L,R,X,S,T)$
7. Return T

$E_2^{-1}(X,S)$ query:
1. If $(2,X,*,S) \in$ history
2. Return $*$
3. $T \leftarrow \mathsf{GuessE}_3(S,X)$
4. $(\mathsf{Err},L,R) \leftarrow \mathsf{Check}^{-1}(S,T,X)$
5. If $\mathsf{Err} =$ True Goto 3
6. $\mathsf{Store}(L,R,X,S,T)$
7. Return R

$\mathsf{Check}^{-1}(S,T,X)$:
1. $L\|R \leftarrow P^{-1}(S\|T)$
2. If $\mathsf{ConflictE}_2(X,R,S) =$ True Or $\mathsf{ConflictE}_1(R,L,X) =$ True
3. Return (True, $*$, $*$)
4. Return (False, L, R)

For a given set of queries Q and their responses $\mathcal{X}(Q)$ we define the *extender consistency* as the property that the responses to Ψ_3 (or P) are equal to those that one would obtain by applying the extender construction from responses to E (or $\mathcal{S}$) (when queries to E or $\mathcal{S}$ suffice to preform the calculation). By construction, the system (Ψ_3, E) always gives consistent response. Also from the definition of our simulator $\mathcal{S}$, it is evident that $(P,\mathcal{S})$ gives consistent responses as well ($\mathcal{S}$ runs in polynomial time unless it enters an exponential loop with a small probability). For any distinguisher $\mathcal{D}$ against the systems (Ψ_3, E) and $(P,\mathcal{S})$ we construct another distinguisher $\mathcal{D}''$ by the following way.

1. $\Psi_3(L,R) \to (S,T)$ query in $\mathcal{D}$ is replaced in $\mathcal{D}''$ by the sequence of queries
$$E_1(R,L) \to X, E_2(X,R) \to S, E_3(S,X) \to T.$$
2. $\Psi_3^{-1}(S,T) \to (L,R)$ query in $\mathcal{D}$ is replaced in $\mathcal{D}''$ by the sequence of queries
$$E_3^{-1}(S,T) \to X, E_2^{-1}(X,S) \to R, E_1^{-1}(R,X) \to L.$$

We argue that $\mathcal{D}''$ is more powerful than $\mathcal{D}$. In replacement 1, $\mathcal{D}''$ actually observes the 5-tuple (L,R,X,S,T), whereas $\mathcal{D}$ can only observe the 4-tuple (L,R,S,T). Moreover, the systems (Ψ_3, E) and $(P,\mathcal{S})$ always give extender-consistent responses. Hence, $\mathcal{D}''$ observes the exact same information as observed by $\mathcal{D}$, plus some extra information. Formally, we have the following theorem,

Theorem 4.5. *For any distinguisher $\mathcal{D}$ and the distinguisher $\mathcal{D}''$ constructed in the above way, we have*

$$\mathsf{Adv}^{\mathcal{D}} \leq \mathsf{Adv}^{\mathcal{D}''}.$$

Moreover, $\mathcal{D}''$ only makes queries to E or $\mathcal{S}$.

Remark 4.4 The above theorem is not true in case of general indifferentiability, this kind of query replacement would mean the simulator is getting some extra information, namely the queries to the P oracle.

For any distinguisher $\mathcal{D}''$ making only E or $\mathcal{S}$ queries we construct another distinguisher D' by the following way.

1. $E_1(R, L) \to X$ query in $\mathcal{D}''$ is replaced by the sequence of queries $E_1(R, L) \to X$, $E_2(X, R) \to S$, $E_3(S, X) \to T$ in $\mathcal{D}'$.
2. $E_1^{-1}(R, X) \to L$ query in $\mathcal{D}''$ is replaced by the sequence of queries $E_1^{-1}(R, X) \to L$, $E_2(X, R) \to S$, $E_3(S, X) \to T$ in $\mathcal{D}'$.
3. $E_2(X, R) \to L$ query in $\mathcal{D}''$ is replaced by the sequence of queries $E_1^{-1}(R, X) \to L$, $E_2(X, R) \to S$, $E_3(S, X) \to T$ in $\mathcal{D}'$.
4. The queries $E_3^{-1}(S, T)$, $E_3(S, X)$ and $E_2^{-1}(X, S)$ are processed essentially symmetrically.
5. $\mathcal{D}'$ does not make any duplicate or *trivial* query. (Queries which try to verify whether E_i is a well defined permutation are trivial queries).

In general, $\mathcal{D}'$ always observes the exact same or more information than $\mathcal{D}''$. It might happen that for some (L, R, S, T) such that $\Psi_3(L, R) = (S, T)$, $\mathcal{D}'$ gets the intermediate X value through $E_1(R, L)$ query, whereas $\mathcal{D}''$ finds it through $E_3^{-1}(S, T)$ query or vice-verse. When the distinguishers are interacting with (Ψ_3, E) the X value is always the same irrespective of whether it is retrieved through E_1 or E_3^{-1} query. And when the distinguishers are interacting with $(P, \mathcal{S})$, X follows the exact same probability distribution in both scenarios. This is due to the fact that, for fixed (L, R, S, T) while answering $E_1(R, L)$ or $E_3^{-1}(S, T)$ the simulator picks X uniformly over all possible X which do not conflict with previous responses. Proof of Lemma 4.12 explains it in more details. Hence we have the following theorem.

Theorem 4.6. *For any distinguisher $\mathcal{D}$ and the distinguishers $\mathcal{D}''$ and $\mathcal{D}'$ (making only E or $\mathcal{S}$ queries) constructed in the above way, we have*

$$\mathsf{Adv}^{\mathcal{D}} \leq \mathsf{Adv}^{\mathcal{D}''} \leq \mathsf{Adv}^{\mathcal{D}'}.$$

Moreover, if $\mathcal{D}$ makes at most q queries to the systems (Ψ_3, E) or $(P, \mathcal{S})$, then $\mathcal{D}'$ only makes at most q many 3-query sequences to (Ψ_3, E) or $(P, \mathcal{S})$. Each query sequence is one of the following types.

- Type I $E_1(R, L) \to X$, $E_2(X, R) \to S$, $E_3(S, X) \to T$

- TYPE II $E_1^{-1}(R, X) \to L$, $E_2(X, R) \to S$, $E_3(S, X) \to T$
- TYPE III $E_3^{-1}(S, T) \to X$, $E_2^{-1}(X, S) \to R$, $E_1^{-1}(R, X) \to L$
- TYPE IV $E_3(S, X) \to T$, $E_2^{-1}(X, S) \to R$, $E_1^{-1}(R, X) \to L$

TYPE I and TYPE III query sequences are actually symmetric. The same is true for TYPE II and IV query sequences as well. We write $\{0,1\}^n$ as $\mathbf{Y}$. If $\mathsf{Adv}_{i+1}^{\mathcal{D}'}$ is the advantage of the distinguisher $\mathcal{D}'$ for the $(i+1)^{th}$ 3-query sequence we have the following two theorems.

Theorem 4.7. *If the $(i+1)^{th}$ 3-query sequence made by $\mathcal{D}'$ is either of* TYPE I *or* TYPE III *we have,*

$$\mathsf{Adv}_{i+1}^{\mathcal{D}'} \leq \frac{2i}{|\mathbf{Y}|^2}.$$

Theorem 4.8. *If the $(i+1)^{th}$ 3-query sequence made by $\mathcal{D}'$ is either of* TYPE II *or* TYPE IV *we have,*

$$\mathsf{Adv}_{i+1}^{\mathcal{D}'} \leq \frac{5i}{|\mathbf{Y}|^2} + \frac{25i^2}{|\mathbf{Y}|^3} + \frac{4i^3}{|\mathbf{Y}|^4}.$$

We know, $\mathsf{Adv}^{\mathcal{D}'} \leq \sum_{i=0}^{q-1} \mathsf{Adv}_{i+1}^{\mathcal{D}'}$. Hence Theorem 4.6, Theorem 4.7 and Theorem 4.8 together complete the proof of Theorem 4.4.

4.9.1 Proof of Theorem 4.7 and Theorem 4.8

Input-output of each 3-query sequence made by $\mathcal{D}'$ is actually a 5-tuple (L, R, X, S, T). In fact, input of each 3-query sequence is a 2-tuple and output is a 3-tuple. Say before $(i+1)^{th}$ query, $\mathcal{D}'$ has observed i many such distinct (as $\mathcal{D}'$ does not make duplicate or trivial queries) 5-tuples $(L_j, R_j, X_j, S_j, T_j)$ for $j = 1$ to i. When $\mathcal{D}'$ is interacting with $(P, \mathcal{S})$, the simulator's internal history also contains exactly the same information. Let $\mathbf{L}$, $\mathbf{R}$, $\mathbf{X}$, $\mathbf{S}$ and $\mathbf{T}$ be the set of L_j's, R_j's, X_j's, S_j's and T_j's (for $j = 1$ to i) respectively. We partition the set $\mathbf{L}$ as

$$\mathbf{L} = \mathbf{L}^1 \cup \mathbf{L}^2 \cup \cdots \cup \mathbf{L}^i,$$

such that $\ell \in \mathbf{L}^k$ if and only if ℓ has appeared exactly k times in history (or there are exactly k-many j values such that $\ell = L_j$). We do similar partitioning for the sets $\mathbf{R}$, $\mathbf{X}$, $\mathbf{S}$ and $\mathbf{T}$ as well. Note,

$$\sum_j j|\mathbf{L}^j| = \sum_j j|\mathbf{R}^j| = \sum_j j|\mathbf{X}^j| = \sum_j j|\mathbf{S}^j| = \sum_j j|\mathbf{T}^j| = i. \tag{4.2}$$

We also define $\mathbf{L}^0 = \mathbf{Y} \setminus \mathbf{L}$. $\mathbf{R}^0$, $\mathbf{X}^0$, $\mathbf{S}^0$ and $\mathbf{T}^0$ are defined similarly. The proofs of Theorem 4.7 and Theorem 4.8 are essentially independent. We describe Theorem 4.8 before, because the proof is simpler.

Theorem 4.8

We prove the result when the query sequence is of Type II. For Type IV query sequence the result follows because of symmetry. Let (R, X) be the input to the Type II query sequence. Note, we can not have $(R, X) = (R_j, X_j)$ for some $j \in [1, i]$, because then $E_1^{-1}(R, X)$ becomes duplicate or trivial query. (L^E, S^E, T^E) and (L^P, S^P, T^P) be the random variables corresponding to the output tuple depending on whether $\mathcal{D}'$ is interacting with (Ψ_3, E) or $(P, \mathcal{S})$ respectively.

Let $\mathbf{B}_L$ be the set of values for which $E_1(R, .)$ is defined in history. Also on $\mathbf{B}_S$, $E_2^{-1}(X, .)$ is defined. If $R \in \mathbf{R}^{i_1}$ and $X \in \mathbf{X}^{i_2}$, we have

$$|\mathbf{B}_L| = i_1 \text{ and } |\mathbf{B}_S| = i_2,$$
$$\mathbf{B}_L \subseteq \mathbf{L} \text{ and } \mathbf{B}_S \subseteq \mathbf{S}.$$

$\mathbf{OP}_{ST}$ be the set of (S_j, T_j) tuples present in history. Also for $j = 1, \cdots, i$, (S_j, T_j)'s are actually distinct. In other words, $|\mathbf{OP}_{ST}| = i$.

Lemma 4.9. *If $R \in \mathbf{R}^{i_1}$ and $X \in \mathbf{X}^{i_2}$ then for $j = 0, \cdots, i$ we have*

$$\Pr[(L^E, S^E, T^E) = (L, S, T)] = \frac{1}{|\mathbf{Y}| - i_1} \times \frac{1}{|\mathbf{Y}| - i_2} \times \frac{1}{|\mathbf{Y}| - j}$$

when $(L, S, T) \in (\mathbf{Y} \setminus \mathbf{B}_L) \times ((\mathbf{S}^j \times \mathbf{Y}) \setminus \mathbf{OP}_{ST})$. Moreover,

$$|(\mathbf{Y} \setminus \mathbf{B}_L) \times (((\mathbf{S}^j \setminus \mathbf{B}_S) \times \mathbf{Y}) \setminus \mathbf{OP}_{ST})| = (|\mathbf{Y}| - i_1)|\mathbf{S}^j \setminus \mathbf{B}_S|(|\mathbf{Y}| - j).$$

For the tuples (L, S, T) not covered by Lemma 4.9, $\Pr[(L^E, S^E, T^E) = (L, S, T)]$ is actually zero. In fact, (L^E, S^E, T^E) always have some non-zero probability over the set $(\mathbf{Y} \setminus \mathbf{B}_L) \times (((\mathbf{Y} \setminus \mathbf{B}_S) \times \mathbf{Y}) \setminus \mathbf{OP}_{ST})$, even though non-uniform. We will see, (L^P, S^P, T^P) is actually uniform over the same set, and at other points it has zero probability as well. Using some basic counting principles, we also get

$$\begin{aligned}|((\mathbf{Y} \setminus \mathbf{B}_S) \times \mathbf{Y}) \setminus \mathbf{OP}_{ST}| &= (|\mathbf{Y}| - |\mathbf{B}_S|)|\mathbf{Y}| - |\mathbf{OP}_{ST}| + \sum_{j'} j'|\mathbf{B}_S \cap \mathbf{S}^{j'}| \\ &= |\mathbf{Y}|^2 - i_2|\mathbf{Y}| - i + \sum_{j'} j'|\mathbf{B}_S \cap \mathbf{S}^{j'}|.\end{aligned}$$

Formally, we have the following lemma.

Lemma 4.10. *If $R \in \mathbf{R}^{i_1}$ and $X \in \mathbf{X}^{i_2}$ then, (L^P, S^P, T^P) is uniform over $(\mathbf{Y} \setminus \mathbf{B}_L) \times (((\mathbf{Y} \setminus \mathbf{B}_S) \times \mathbf{Y}) \setminus \mathbf{OP}_{ST})$. More specifically, for $(L, S, T) \in (\mathbf{Y} \setminus \mathbf{B}_L) \times (((\mathbf{Y} \setminus \mathbf{B}_S) \times \mathbf{Y}) \setminus \mathbf{OP}_{ST})$ we have,*

$$\Pr[(L^P, S^P, T^P) = (L, S, T)] = \frac{1}{|\mathbf{Y}| - i_1} \times \frac{1}{|\mathbf{Y}|^2 - i_2|\mathbf{Y}| - i + \sum_{j'} j'|\mathbf{B}_S \cap \mathbf{S}^{j'}|}.$$

Proof. For a Type II query sequence the simulator response is decided by simulators behavior on $E_1^{-1}(R, X)$ query. Inside the $\mathsf{Check}(L, R, X)$ function, L is not passed to $\mathsf{ConflictE_2}$ and $\mathsf{ConflictE_3}$ functions. L, is only used as input to P. P

being a random permutation the probability distributions of L^P and (S^P, T^P) are actually independent. Again, due to uniform randomness of P, (S^P, T^P) is actually uniform over a set of possible values which does conflict with history. Again, the probability that $\mathsf{Check}(L, R, X)$ returns Err is the same for all possible outputs of GuessE_1^{-1}. Also GuessE_1^{-1} outputs uniformly. Hence, the distribution of L^E is also uniform. The result follows, because joint probability distribution of two uniform and independent distribution is also uniform. □ □

$\mathsf{Adv}_{i+1}^{\mathcal{D}'}$ is nothing but sum of half of the probability differences of (L^E, S^E, T^E) and (L^P, S^P, T^P) at all points. Instead of considering all the points we can only consider the points where the probability corresponding to (L^P, S^P, T^P) is bigger. Now let us calculate the probability differences. For $j = 0, \cdots, i$ if $(L, S, T) \in (\mathbf{Y} \setminus \mathbf{B}_L) \times ((\mathbf{S}^j \times \mathbf{Y}) \setminus \mathbf{OP}_{ST})$ we have,

$$\begin{aligned}
&\Pr[(L^E, S^E, T^E) = (L, S, T)] - \Pr[(L^P, S^P, T^P) = (L, S, T)] \\
&= \frac{1}{|\mathbf{Y}| - i_1} \times \frac{1}{|\mathbf{Y}| - i_2} \times \frac{1}{|\mathbf{Y}| - j} - \frac{1}{|\mathbf{Y}| - i_1} \times \frac{1}{|\mathbf{Y}|^2 - i_2|\mathbf{Y}| - i + \sum_{j'} j'|\mathbf{B}_S \cap \mathbf{S}^{j'}|} \\
&= \frac{j(|\mathbf{Y}| - i_2) - i + \sum_{j'} j'|\mathbf{B}_S \cap \mathbf{S}^{j'}|}{(|\mathbf{Y}| - i_1)(|\mathbf{Y}| - i_2)(|\mathbf{Y}| - j)(|\mathbf{Y}|^2 - i_2|\mathbf{Y}| - i + \sum_{j'} j'|\mathbf{B}_S \cap \mathbf{S}^{j'}|)}
\end{aligned}$$

- As, $i \leq |\mathbf{Y}|/2$ and $i_1 \leq i$ the expression above is bigger than zero for $j \geq 1$.
- Also, $\sum_{j'} j'|\mathbf{B}_S \cap \mathbf{S}^{j'}| \leq \sum_{j'} j'|\mathbf{S}^{j'}| \leq i$. Hence the expression above is negative for $j = 0$.

So, we can only consider $j = 0$, for calculating $\mathsf{Adv}_{i+1}^{\mathcal{D}'}$.

$$\begin{aligned}
\mathsf{Adv}_{i+1}^{\mathcal{D}'} &\leq \frac{(|\mathbf{Y}| - i_1)(|\mathbf{Y}| - |\mathbf{S}|)|\mathbf{Y}| \times i}{(|\mathbf{Y}| - i_1)(|\mathbf{Y}| - i_2)|\mathbf{Y}|(|\mathbf{Y}|^2 - i_2|\mathbf{Y}| - i + \sum_{j'} j'|\mathbf{B}_S \cap \mathbf{S}^{j'}|)} \\
&\leq \frac{i}{|\mathbf{Y}|^2 - i|\mathbf{Y}| - i} \qquad (As,\ i_2 = |\mathbf{B}_S| \leq |\mathbf{S}| \text{ and } i_2 \leq i) \\
&\leq \frac{2i}{|\mathbf{Y}|^2} \qquad (As,\ i \leq |\mathbf{Y}|/2 - 1)
\end{aligned}$$

Theorem 4.7

We prove the result when the query sequence is of Type I. For Type III query sequence the result follows because of symmetry. Let (L, R) be the input to the Type I query sequence. Note, we can not have $(L, R) = (L_j, R_j)$ for some $j \in [1, i]$, because then $E_1(R, L)$ becomes duplicate or trivial query. (X^E, S^E, T^E) and (X^P, S^P, T^P) be the random variables corresponding to the output tuple depending on whether $\mathcal{D}'$ is interacting with (Ψ_3, E) or $(P, \mathcal{S})$ respectively.

As before, $\mathbf{B}_X$ is the set of values for which $E_1^{-1}(R, .)$ is already defined in history. Let us assume $R \in \mathbf{R}^{i_1}$. We have

$$|\mathbf{B}_X| = i_1 \text{ and } \mathbf{B}_X \subseteq \mathbf{X}.$$

We also partition the sets $\mathbf{S}^j$'s as follows. For $j = 0, \cdots, i$,

$$\mathbf{S}^j = \mathbf{S}^j_0 \cup \mathbf{S}^j_1 \cup \cdots \cup \mathbf{S}^j_{\min(i_1,j)},$$

such that $s \in \mathbf{S}^j_{j'}$ if and only if there are exactly j' many $x \in \mathbf{B}_X$ for which $E_3^{-1}(s, x)$ is defined in history. $|\mathbf{S}^j_{j'}| > 0$ actually implies,

$$i_1 + j - j' \leq i.$$

$\mathbf{OP}_{XS}$ be the set of (X_j, S_j) tuples present in history. We have,

$$|\mathbf{OP}_{XS}| = i.$$

$\mathbf{OP}_{ST}$ is defined as before. Let, us denote $|\mathbf{X}^k \cap \mathbf{B}_X|$ as r_k. Note,

$$\sum_{k=1}^{i} r_k = |\mathbf{B}_X| = i_1 \text{ and } \sum_{k=1}^{i} k r_k \geq i_1.$$

Lemma 4.11. *If $R \in \mathbf{R}^{i_1}$, then for $k = 0, \cdots, i$ and $j = 0, \cdots, i$ we have*

$$\Pr[(X^E, S^E, T^E) = (X, S, T)] = \frac{1}{|\mathbf{Y}| - i_1} \times \frac{1}{|\mathbf{Y}| - k} \times \frac{1}{|\mathbf{Y}| - j},$$

when $(X, S, T) \in ((\mathbf{X}^k \setminus \mathbf{B}_X) \times \mathbf{S}^j \times \mathbf{Y}) \setminus ((\mathbf{OP}_{XS} \times \mathbf{Y}) \cup (\mathbf{Y} \times \mathbf{OP}_{ST}))$.

Lemma 4.12. *If $R \in \mathbf{R}^{i_1}$, then for $j = 0, \cdots, i$ and $j' = 0, \cdots, \min(i_1, j)$ we have*

$$\Pr[(X^P, S^P, T^P) = (X, S, T)] = \frac{1}{|\mathbf{Y}| - i_1 - j + j'} \times \frac{1}{|\mathbf{Y}|^2 - i},$$

when $(X, S, T) \in ((\mathbf{Y} \setminus \mathbf{B}_X) \times \mathbf{S}^j_{j'} \times \mathbf{Y}) \setminus ((\mathbf{OP}_{XS} \times \mathbf{Y}) \cup (\mathbf{Y} \times \mathbf{OP}_{ST}))$.

Proof. For a TYPE I query sequence the simulator response is decided by simulators behavior on $E_1(R, L)$ query. (S, T) values return by the simulator is actually direct output of P. Hence, the distribution of (S^E, T^E) is independent of internal random choices of the simulator. If we fix S then the distribution of X^E is actually uniform over the values for which $E_1^{-1}(R, .)$ and $E_3(S, .)$ is not defined. (Note, we can actually drop the $\mathsf{ConflictE}_2$ call inside $\mathsf{Check}(L, R, X)$ function when it is being called from E_1). □ □

Note, both (X^E, S^E, T^E) and (X^P, S^P, T^P) has non-zero probabilities over same set of points (this is a consequence of our simulator being always consistent to P, and never aborting), although they are not the same. $\mathsf{Adv}^{\mathcal{D}'}_{i+1}$ is nothing but half of sum of the probability differences of the two distributions over all points. The distribution of $(X^E, S^E, T^)$ does not depend on j', where as the distribution of (X^P, S^P, T^P) does not depend on k. To give an upper bound for the sum of probability differences we divide the total probability space in four parts. Then, we give an upper bound for each part separately.

1. Δ_{00} is the sum of probability differences for the points, where $j = 0$ and $k = 0$. Formally,

$$\Delta_{00} = \sum_{\substack{(X,S,T)\in \\ (\mathbf{Y}\setminus\mathbf{X})\times(\mathbf{Y}\setminus\mathbf{S})\times\mathbf{Y}}} \big|\Pr[(X^E,S^E,T^E)=(X,S,T)] - \Pr[(X^P,S^P,T^P)=(X,S,T)]\big|$$

2. Δ_{01} is the sum of probability differences for the points where, $j = 0$ and $k \geq 1$. Formally,

$$\Delta_{01} = \sum_{k=1}^{i} \sum_{\substack{(X,S,T)\in \\ (\mathbf{X}^k\setminus\mathbf{B}_X)\times(\mathbf{Y}\setminus\mathbf{S})\times\mathbf{Y}}} \big|\Pr[(X^E,S^E,T^E)=(X,S,T)] - \Pr[(X^P,S^P,T^P)=(X,S,T)]\big|$$

3. Δ_{10} is the sum of probability differences for the points where, $j \geq 1$ and $k = 0$. Formally,

$$\Delta_{01} = \sum_{j=1}^{i} \sum_{j'=0}^{\min(i_1,j)} \sum_{\substack{(X,S,T)\in \\ (\mathbf{Y}\setminus\mathbf{X})\times((\mathbf{S}^j_{j'}\times\mathbf{Y})\setminus\mathbf{OP}_{ST})}} \big|\Pr[(X^E,S^E,T^E)=(X,S,T)] - \Pr[(X^P,S^P,T^P)=(X,S,T)]\big|$$

4. Δ_{11} is the sum of probability differences for the points where, $j \geq 1$ and $k \geq 1$. Formally,

$$\Delta_{11} = \sum_{k=1}^{i}\sum_{j=1}^{i} \sum_{j'=0}^{\min(i_1,j)} \sum_{\substack{(X,S,T)\in \\ ((\mathbf{X}^k\setminus\mathbf{B}_X)\times\mathbf{S}^j_{j'}\times\mathbf{Y}) \\ \setminus((\mathbf{OP}_{XS}\times\mathbf{Y})\cup(\mathbf{Y}\times\mathbf{OP}_{ST}))}} \big|\Pr[(X^E,S^E,T^E)=(X,S,T)] - \Pr[(X^P,S^P,T^P)=(X,S,T)]\big|$$

Below, we state the upper bounds for Δ_{ij}'s. In section 4.9.2 we give a detailed analysis.

1. $\Delta_{00} \leq \frac{2i}{|\mathbf{Y}|^2}$
2. $\Delta_{01} \leq \frac{4i}{|\mathbf{Y}|^2}$
3. $\Delta_{10} \leq \frac{4i}{|\mathbf{Y}|^2} + \frac{10i^2}{|\mathbf{Y}|^3}$
4. $\Delta_{11} \leq \frac{40i^2}{|\mathbf{Y}|^3} + \frac{8i^3}{|\mathbf{Y}|^4}$

Hence,

$$\begin{aligned}\mathsf{Adv}_{i+1}^{\mathcal{D}'} &= \frac{1}{2}(\Delta_{00}+\Delta_{01}+\Delta_{10}+\Delta_{11}) \\ &\leq \frac{5i}{|\mathbf{Y}|^2}+\frac{25i^2}{|\mathbf{Y}|^3}+\frac{4i^3}{|\mathbf{Y}|^4}\end{aligned}$$

4.9.2 Upper bound for Δ_{ij}'s

Upper bound for Δ_{00}

If $(X,S,T) \in (\mathbf{Y}\setminus\mathbf{X})\times(\mathbf{Y}\setminus\mathbf{S})\times\mathbf{Y}$, we have

$$\begin{aligned}&|\Pr[(X^E,S^E,T^E)=(X,S,T)]-\Pr[(X^P,S^P,T^P)=(X,S,T)]| \\ &= |\frac{1}{|\mathbf{Y}|-i_1}\times\frac{1}{|\mathbf{Y}|^2}-\frac{1}{|\mathbf{Y}|-i_1}\times\frac{1}{|\mathbf{Y}|^2-i}| = \frac{i}{|\mathbf{Y}|^2(|\mathbf{Y}|-i_1)(|\mathbf{Y}|^2-i)}.\end{aligned}$$

Hence,

$$\begin{aligned}\Delta_{00} &= \frac{(|\mathbf{Y}|-|\mathbf{X}|)(|\mathbf{Y}|-|\mathbf{S}|)|\mathbf{Y}|\times i}{|\mathbf{Y}|^2(|\mathbf{Y}|-i_1)(|\mathbf{Y}|^2-i)} \\ &= \frac{i}{|\mathbf{Y}|^2-i}\times\frac{|\mathbf{Y}|-|\mathbf{S}|}{|\mathbf{Y}|}\times\frac{|\mathbf{Y}|-|\mathbf{X}|}{|\mathbf{Y}|-i_1} \\ &\leq \frac{i}{|\mathbf{Y}|^2-i} \qquad (As,\, i_1=|\mathbf{B}_X|\leq|\mathbf{X}|) \\ &\leq \frac{2i}{|\mathbf{Y}|^2}\end{aligned}$$

Upper bound for Δ_{01}

If $(X,S,T) \in (\mathbf{X}^k\setminus\mathbf{B}_X)\times(\mathbf{Y}\setminus\mathbf{S})\times\mathbf{Y}$, we have

$$\begin{aligned}&|\Pr[(X^E,S^E,T^E)=(X,S,T)]-\Pr[(X^P,S^P,T^P)=(X,S,T)]| \\ &= |\frac{1}{|\mathbf{Y}|-i_1}\times\frac{1}{|\mathbf{Y}|-k}\times\frac{1}{|\mathbf{Y}|}-\frac{1}{|\mathbf{Y}|-i_1}\times\frac{1}{|\mathbf{Y}|^2-i}| \\ &= \frac{k|\mathbf{Y}|-i}{|\mathbf{Y}|(|\mathbf{Y}|-i_1)(|\mathbf{Y}|-k)(|\mathbf{Y}|^2-i)}.\end{aligned}$$

Observe,

$$|(\mathbf{X}^k\setminus\mathbf{B}_X)\times(\mathbf{Y}\setminus\mathbf{S})\times\mathbf{Y}| = (|\mathbf{X}^k|-r_k)(|\mathbf{Y}|-|\mathbf{S}|)|\mathbf{Y}|.$$

Hence,

$$\begin{aligned}
\Delta_{01} &= \sum_{k=1}^{i} \frac{(|\mathbf{X}^k| - r_k)(|\mathbf{Y}| - |\mathbf{S}|)|\mathbf{Y}|(k|\mathbf{Y}| - i)}{|\mathbf{Y}|(|\mathbf{Y}| - i_1)(|\mathbf{Y}| - k)(|\mathbf{Y}|^2 - i)} \\
&\leq \frac{(|\mathbf{Y}| - |\mathbf{S}|)}{(|\mathbf{Y}| - i_1)(|\mathbf{Y}| - i)(|\mathbf{Y}|^2 - i)} \times \sum_{k=1}^{i} (|\mathbf{X}^k| - r_k)(k|\mathbf{Y}| - i) \qquad (As,\ k \leq i) \\
&\leq \frac{i(|\mathbf{Y}| - |\mathbf{X}|)(|\mathbf{Y}| - |\mathbf{S}|))}{(|\mathbf{Y}| - i_1)(|\mathbf{Y}| - i)(|\mathbf{Y}|^2 - i)} \\
&\qquad\qquad (As,\ \sum_{k=1}^{i} k|\mathbf{X}^k| = i, \sum_{k=1}^{i} |\mathbf{X}^k| = |\mathbf{X}|, \sum_{k=1}^{i} r_k = i_1, \sum_{k=1}^{i} kr_k \geq i_1 \text{ and } |\mathbf{Y}| \geq i) \\
&\leq \frac{i|\mathbf{Y}|}{(|\mathbf{Y}| - i)(|\mathbf{Y}|^2 - i)} \qquad (As,\ i_1 \leq |\mathbf{X}|) \\
&\leq \frac{4i}{|\mathbf{Y}|^2} \qquad (As,\ i \leq |\mathbf{Y}|/2)
\end{aligned}$$

Upper bound for Δ_{10}

If $(X, S, T) \in (\mathbf{Y} \setminus \mathbf{X}) \times ((\mathbf{S}^j_{j'} \times \mathbf{Y}) \setminus \mathbf{OP}_{ST})$, we have

$$\begin{aligned}
&|\Pr[(X^E, S^E, T^E) = (X, S, T)] - \Pr[(X^P, S^P, T^P) = (X, S, T)]| \\
&= \Big|\frac{1}{|\mathbf{Y}| - i_1} \times \frac{1}{|\mathbf{Y}|} \times \frac{1}{|\mathbf{Y}| - j} - \frac{1}{|\mathbf{Y}| - i_1 - j + j'} \times \frac{1}{|\mathbf{Y}|^2 - i}\Big| \\
&= \Big|\frac{j'|\mathbf{Y}|^2 - (i_1 j + i)|\mathbf{Y}| + i(i_1 + j - j')}{|\mathbf{Y}|(|\mathbf{Y}| - i_1)(|\mathbf{Y}| - j)(|\mathbf{Y}| - i_1 - j + j')(|\mathbf{Y}|^2 - i)}\Big| \\
&\leq \frac{j'|\mathbf{Y}|^2 + (i_1 j + i)|\mathbf{Y}| + i(i_1 + j - j')}{|\mathbf{Y}|(|\mathbf{Y}| - i_1)(|\mathbf{Y}| - j)(|\mathbf{Y}| - i)(|\mathbf{Y}|^2 - i)} \qquad (As,\ j' \leq \min(i_1, j) \text{ and } i_1 + j - j' \leq i).
\end{aligned}$$

Also,

$$|(\mathbf{Y} \setminus \mathbf{X}) \times ((\mathbf{S}^j_{j'} \times \mathbf{Y}) \setminus \mathbf{OP}_{ST})| = (|\mathbf{Y}| - |\mathbf{X}|)|\mathbf{S}^j_{j'}|(|\mathbf{Y}| - j).$$

Hence,

$$\begin{aligned}
\Delta_{10} &\leq \sum_{j=1}^{i} \sum_{j'=0}^{\min(i_1,j)} \frac{(|\mathbf{Y}|-|\mathbf{X}|)|\mathbf{S}^j_{j'}|(j'|\mathbf{Y}|^2+(i_1j+i)|\mathbf{Y}|+i(i_1+j-j'))}{|\mathbf{Y}|(|\mathbf{Y}|-i_1)(|\mathbf{Y}|-i)(|\mathbf{Y}|^2-i)} \\
&\leq \sum_{j=1}^{i} \frac{(|\mathbf{Y}|-|\mathbf{X}|)(j|\mathbf{S}^j||\mathbf{Y}|^2+(i_1j|\mathbf{S}^j|+i|\mathbf{S}^j|)|\mathbf{Y}|+(ii_1+j)|\mathbf{S}^j|)}{|\mathbf{Y}|(|\mathbf{Y}|-i_1)(|\mathbf{Y}|-i)(|\mathbf{Y}|^2-i)} \\
&\qquad (As,\ j' \leq j \ and \sum_{j'=0}^{\min(j,i_1)} |\mathbf{S}^j_{j'}| = |\mathbf{S}^j|) \\
&\leq \frac{i|\mathbf{Y}|^2+(i_1+|\mathbf{S}|)i|\mathbf{Y}|+i(i_1|\mathbf{S}|+1)}{|\mathbf{Y}|(|\mathbf{Y}|-i)(|\mathbf{Y}|^2-i)} \qquad (As,\ \sum_{j=1}^{i}|\mathbf{S}^j| = |\mathbf{S}|, \sum_{j=1}^{i} j|\mathbf{S}^j| = i \ and \ i_1 \leq |\mathbf{X}|) \\
&\leq \frac{i|\mathbf{Y}|^2+2i^2|\mathbf{Y}|+(i^3+i^2)}{|\mathbf{Y}|(|\mathbf{Y}|-i)(|\mathbf{Y}|^2-i)} \qquad (As,\ i_1 \leq i \ and \ |\mathbf{S}| \leq i) \\
&\leq \frac{4i}{|\mathbf{Y}|^2} + \frac{10i^2}{|\mathbf{Y}|^3} \qquad (As,\ i \leq |\mathbf{Y}|/2-1)
\end{aligned}$$

Upper bound for Δ_{11}

If $(X,S,T) \in ((\mathbf{Y} \setminus \mathbf{B}_X) \times \mathbf{S}^j_{j'} \times \mathbf{Y}) \setminus ((\mathbf{OP}_{XS} \times \mathbf{Y}) \cup (\mathbf{Y} \times \mathbf{OP}_{ST}))$, we have

$$\begin{aligned}
&|\Pr[(X^E,S^E,T^E)=(X,S,T)] - \Pr[(X^P,S^P,T^P)=(X,S,T)]| \\
&= |\frac{1}{|\mathbf{Y}|-i_1} \times \frac{1}{|\mathbf{Y}|-k} \times \frac{1}{|\mathbf{Y}|-j} - \frac{1}{|\mathbf{Y}|-i_1-j+j'} \times \frac{1}{|\mathbf{Y}|^2-i}| \\
&= |\frac{(k+j')|\mathbf{Y}|^2-(i+i_1k+kj+i_1j)|\mathbf{Y}|+(ii_1+ij-ij'+i_1kj)}{(|\mathbf{Y}|-i_1)(|\mathbf{Y}|-k)(|\mathbf{Y}|-j)(|\mathbf{Y}|-i_1-j+j')(|\mathbf{Y}|^2-i)}| \\
&\leq \frac{(k+j')|\mathbf{Y}|^2+(i+i_1k+kj+i_1j)|\mathbf{Y}|+(ii_1+ij-ij'+i_1kj)}{(|\mathbf{Y}|-i_1)(|\mathbf{Y}|-k)(|\mathbf{Y}|-j)(|\mathbf{Y}|-i)(|\mathbf{Y}|^2-i)} \\
&\qquad (As,\ j' \leq \min(i_1,j) \ and \ i_1+j-j' \leq i).
\end{aligned}$$

Also, note

$$\begin{aligned}
|((\mathbf{X}^k \setminus \mathbf{B}_X) \times \mathbf{S}^j_{j'} \times \mathbf{Y}) \setminus ((\mathbf{OP}_{XS} \times \mathbf{Y}) \cup (\mathbf{Y} \times \mathbf{OP}_{ST}))| &\leq |((\mathbf{X}^k \setminus \mathbf{B}_X) \times \mathbf{S}^j_{j'} \times \mathbf{Y}) \setminus (\mathbf{Y} \times \mathbf{OP}_{ST})| \\
&= (|\mathbf{X}^k| - r_k)|\mathbf{S}^j_{j'}|(|\mathbf{Y}|-j).
\end{aligned}$$

Hence,

$$\Delta_{11} \leq \sum_{k=1}^{i}\sum_{j=1}^{i}\sum_{j'=0}^{\min(i_1,j)} (|\mathbf{X}^k| - r_k)|\mathbf{S}^j_{j'}|(|\mathbf{Y}| - j)$$
$$\times \frac{(k+j')|\mathbf{Y}|^2 + (i + i_1k + kj + i_1j)|\mathbf{Y}| + (ii_1 + ij - ij' + i_1kj)}{(|\mathbf{Y}| - i_1)(|\mathbf{Y}| - k)(|\mathbf{Y}| - j)(|\mathbf{Y}| - i)(|\mathbf{Y}|^2 - i)}$$
$$\leq \sum_{k=1}^{i}\sum_{j=1}^{i}(|\mathbf{X}^k| - r_k) \times \frac{(k|\mathbf{S}^j| + j|\mathbf{S}^j|)|\mathbf{Y}|^2 + (i + i_1k + kj + i_1j)|\mathbf{S}^j||\mathbf{Y}| + (ii_1 + ij + i_1kj)|\mathbf{S}^j|}{(|\mathbf{Y}| - i_1)(|\mathbf{Y}| - k)(|\mathbf{Y}| - i)(|\mathbf{Y}|^2 - i)}$$
$$(As,\ j' \leq j\ and \sum_{j'=0}^{\min(i_1,j)} |\mathbf{S}^j_{j'}| = |\mathbf{S}^j|)$$
$$= \sum_{k=1}^{i}(|\mathbf{X}^k| - r_k) \times \frac{(k|\mathbf{S}| + i)|\mathbf{Y}|^2 + (i|\mathbf{S}| + i_1k|\mathbf{S}| + ki + i_1i)|\mathbf{Y}| + (ii_1|\mathbf{S}| + i^2 + ii_1k)}{(|\mathbf{Y}| - i_1)(|\mathbf{Y}| - i)^2(|\mathbf{Y}|^2 - i)}$$
$$(As,\ k \leq i, \sum_{j=1}^{i} |\mathbf{S}^j| = |\mathbf{S}|\ and \sum_{j=1}^{i} j|\mathbf{S}^j| = i)$$
$$\leq \frac{i(|\mathbf{S}| + |\mathbf{X}|)|\mathbf{Y}|^2 + (i|\mathbf{X}||\mathbf{S}| + i^2 + ii_1|\mathbf{X}|)|\mathbf{Y}| + ii_1|\mathbf{S}||\mathbf{X}| + i^2|\mathbf{X}|}{(|\mathbf{Y}| - i_1)(|\mathbf{Y}| - i)^2(|\mathbf{Y}|^2 - i)}$$
$$(As,\ \sum_{k=1}^{i} |\mathbf{X}^k| = |\mathbf{X}|\ and \sum_{k=1}^{i} k|\mathbf{X}^k| = i)$$
$$\leq \frac{2i^2|\mathbf{Y}|^2 + (2i^3 + i^2)|\mathbf{Y}| + (i^4 + i^3)}{(|\mathbf{Y}| - i)^3(|\mathbf{Y}|^2 - i)} \qquad (As,\ |\mathbf{S}| \leq i, |\mathbf{X}| \leq i\ and\ i_1 \leq i)$$
$$\leq \frac{40i^2}{|\mathbf{Y}|^3} + \frac{8i^3}{|\mathbf{Y}|^4} \qquad (As,\ i \leq |\mathbf{Y}|/2 - 1)$$

4.10 Conclusion

We have shown, the 6-round Feistel construction with random round functions is publicly indifferentiable from a random invertible permutation. In general indifferentiability model the best known result is 14-rounds [HKT11]. We introduce a new notion of indifferentiability called sequential indifferentiability, which is a conceptually simpler version of public indifferentiability. However, these two notions are equivalent for stateless ideal primitives. We have also shown sequential indifferentiability implies correlation intractability. This also rules out a wide class of attacks (which utilizes evasive relations) against the 6-round Feistel construction in general indifferentiability model. However, whether the 6-round Feistel construction is indifferentiable from an invertible random permutation or not; still remains an open problem.

Chapter 5

PSS is Secure against Random Fault Attacks

A fault attack consists in inducing hardware malfunctions in order to recover secrets from electronic devices. One of the most famous fault attack is Bellcore's attack against RSA with CRT; it consists in inducing a fault modulo p but not modulo q at signature generation step; then by taking a gcd the attacker can recover the factorization of $N = pq$. The Bellcore attack applies to any encoding function that is deterministic, for example FDH. Recently, the attack was extended to *randomized* encodings based on the ISO/IEC 9796-2 signature standard. Extending the attack to other randomized encodings remains an open problem. In this chapter, we show that the Bellcore attack cannot be applied to the PSS encoding; namely we show that PSS is provably secure against random fault attacks in the random oracle model, assuming that inverting RSA is hard. This is a joint work with Jean-Sébastien Coron [CM09].

Contents

5.1 Introduction

RSA [RSA78] is still the most widely used signature scheme in practical applications. To sign a message m with RSA, the signer first applies an encoding function μ to m, and then computes the signature $\sigma = \mu(m)^d \bmod N$. The signature is verified by checking that $\sigma^e = \mu(m) \mod N$. For efficiency reasons RSA signatures are often computed using the Chinese Remainder Theorem (CRT); in this case the signature is first computed modulo p and q separately:

$$\sigma_p = m^d \mod p\,, \qquad \sigma_q = m^d \mod q$$

and then σ_p and σ_q are combined by CRT to form the signature σ.

Boneh, DeMillo and Lipton showed that RSA signatures computed with CRT can be vulnerable to fault attacks [BDL97, BDL01]. If the attacker can induce a fault when σ_q is computed while keeping the computation of σ_p correct, one obtains:

$$\sigma_p = m^d \mod p\,, \qquad \sigma_q \neq m^d \mod q$$

and the resulting faulty signature σ satisfies

$$\sigma^e = m \mod p\,, \qquad \sigma^e \neq m \mod q\,.$$

Therefore, given one faulty signature σ, the attacker can recover the factorization of N by computing $\gcd(\sigma^e - m \bmod N, N) = p$. This attack actually applies to any deterministic RSA encoding, *e.g.* Full Domain Hash (FDH) [BR96] with $\sigma = H(m)^d \bmod N$.

More generally, the attack applies to any probabilistic scheme where the random used to generate the signature is sent along with the signature, *e.g.* as in the Probabilistic Full Domain Hash (Coron02) encoding [Cor02] where the signature is $\sigma \| r$ with $\sigma = H(m \,\|\, r)^d \bmod N$. In that case, given the faulty value of σ and knowing r, the attacker can still factor N by computing $\gcd(\sigma^e - H(m \,\|\, r) \bmod N, N) = p$.

However, if the random r is not given to the attacker along with the signature σ then the Bellcore attack is thwarted. This is the case for signatures of the form $\sigma = \mu(m, r)^d \bmod N$ where the random r is only recovered when verifying the signature, as in PSS [BR96]. To recover r one needs a *correct* signature; from a faulty signature, the attacker cannot retrieve r nor infer $\mu(m, r)$ in order to compute $\gcd(\sigma^e - \mu(m, r) \bmod N, N) = p$, unless r is short enough to be guessed by exhaustive search. Note that obtaining another correct signature for m would not help the attacker since with high probability a different random r' would be used to generate this signature.

Recently, it was shown how to extend Bellcore's attack to a large class of randomized RSA encoding schemes [CJK+09]. The extended attack was illustrated with the ISO/IEC 9796-2 standard [ISO02]. ISO/IEC 9796-2 is originally a deterministic

encoding scheme but often used in combination with message randomization, as in the EMV standard [EMV08]. The ISO/IEC 9796-2 encoded message has the form

$$\mu(m) = \texttt{6A}_{16} \,\|\, m[1] \,\|\, H(m) \,\|\, \texttt{BC}_{16}$$

where $m = m[1] \,\|\, m[2]$ is split into two parts. The authors of [CJK+09] showed that if the randomness introduced into $m[1]$ is not too large (*e.g.* less than 160 bits for a 2048-bit RSA modulus), then a single faulty signature allows to factor N as in the original Bellcore attack. The attack is based on Coppersmith's technique for finding small roots of polynomial equations [Cop97], which is based on the LLL algorithm [LLL82].

However, extending the attack to other randomized RSA signatures remains an open problem. In particular, it is natural to ask whether the Bellcore attack could apply to PSS [BR96], the most popular RSA-based signature scheme. In this chapter, we show that the Bellcore attack cannot be extended to PSS; namely we show that PSS is provably secure against random fault attacks in the random oracle model, assuming that inverting RSA is hard.

More precisely, we consider an extended model of security in which the attacker, in addition to the regular signing oracle, has access to a faulty signature oracle; that is, the attacker can request faulty signatures either modulo p or modulo q. For a faulty signature modulo q, the signer first generates the correct value modulo p:

$$\sigma_p = \mu(m, r)^d \mod p$$

but generates a random σ_q modulo q. With CRT the signer then computes σ' such that $\sigma' = \sigma_p \mod p$ and $\sigma' = \sigma_q \mod q$, and returns the faulty signature σ' to the adversary. Our result is that PSS is still secure under this extended notion of security, in the random oracle model, assuming that inverting RSA is hard.

5.2 Security Model

We recall the definition of a signature scheme.

Definition 5.1 (signature scheme)**.** *A signature scheme* $(\texttt{Gen}, \texttt{Sign}, \texttt{Verify})$ *is defined as follows:*

- The key generation algorithm `Gen` *is a probabilistic algorithm which given* 1^k*, outputs a pair of matching public and private keys,* (pk, sk)*.*

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

- The verification algorithm `Verify` *takes a message* M*, a candidate signature* x' *and* pk*. It returns a bit* $\texttt{Verify}_{pk}(M, x')$*, equal to one if the signature is accepted, and zero otherwise. We require that if* $x \leftarrow \texttt{Sign}_{sk}(M)$*, then* $\texttt{Verify}_{pk}(M, x) = 1$*.*

In the *existential unforgeability under an adaptive chosen message attack* scenario, the forger can dynamically obtain signatures of messages of his choice and attempts to output a valid forgery. A *valid forgery* is a message/signature pair

(M, x) such that $\texttt{Verify}_{pk}(M, x) = 1$ whereas the signature of M was never requested by the forger.

In the following, we consider an extended model of security in which the attacker, in addition to the regular signing oracle, has access to a faulty signature oracle; that is, the attacker can request faulty signatures either modulo p or modulo q. For a faulty signature modulo q, the signer first generates the correct value modulo p:

$$\sigma_p = \mu(m, r)^d \mod p$$

and generates a random σ_q modulo q. With CRT the signer then computes σ' such that $\sigma' = \sigma_p \mod p$ and $\sigma' = \sigma_q \mod q$, and returns the faulty signature σ' to the adversary. This is actually equivalent to first computing a correct signature σ:

$$\sigma = \mu(m, r)^d \mod N$$

and then generating a random u modulo q and computing the faulty signature:

$$\sigma' = \sigma + u \cdot p \mod N$$

Formally, we consider the following scenario between a challenger and an attacker. Our scenario applies to any RSA based signature scheme in which a signature σ is computed as $\sigma = \mu(m, r)^d \mod N$ for some (randomized) encoding function $\mu(m, r)$.

Setup: the challenger generates an RSA modulus $N = p \cdot q$, a public exponent e such that $\gcd(e, \phi(N)) = 1$ and a private exponent d such that $e \cdot d = 1 \mod \phi(N)$. The challenger sends (N, e) to the adversary.

Queries: the adversary can make regular signature queries to the challenger. In this case, given a message m, the challenger generates a random r and output the (correct) signature:

$$\sigma = \mu(m, r)^d \mod N$$

Additionally, the attacker can make faulty signature queries. For every such query, the attacker specifies whether the fault should be modulo p or modulo q. For a faulty signature modulo q, the challenger first generates a random r and computes the correct signature:

$$\sigma = \mu(m, r)^d \mod N$$

Then the challenger generates a random u modulo q, and computes:

$$\sigma' = \sigma + u \cdot p \mod N$$

and sends σ' to the attacker. The challenger proceeds similarly if a faulty signature modulo p is requested.

Forgery: eventually the attacker must output a forgery, that is a message signature pair (m, x) such that $\texttt{Verify}_{pk}(m, x) = 1$ whereas the signature of m was never requested by the forger, neither as a regular signature query nor in a faulty signature query.

This completes the description of the attack scenario. As usual, we say that a signature scheme is (t, ε)-secure if no adversary running in time t can output a forgery with probability better than ε.

The PSS scheme was proven secure in the random oracle model [BR93], and our security proof with faulty signatures is also in the random oracle model. It is well known that a security proof in the random oracle model does not necessarily imply that a scheme is secure in the real world (see [CGH04]). Although it is always better to have a security proof in the standard model, we think that it is still better to have a proof in the random oracle model than no proof at all.

5.2.1 Why Random Faults ?

In our security model we have assumed that when a faulty signature σ' is obtained, it has the uniform distribution modulo p (or modulo q). This could be seen as a very strong assumption; namely in practice the faults might have a completely non-random distribution. Consider for example a fault attack inducing the values of the registers to be set to zero. This gives $\sigma_p = 0$ and recovering p is then straightforward: simply compute $\gcd(\sigma', N) = p$. To prevent from this attack we could assume that when a fault occurs the value σ_p still has enough min-entropy.

In the following we argue that 1) the random fault assumption is almost unavoidable if we want to obtain a security proof and 2) such assumption might actually be reasonable in practice.

Assume that a fault gives a random $\sigma_p \mod p$ but with the k most significant bits set to 0, for some small integer k. That is, the attacker can obtain a list of faulty signatures σ'_i such that the corresponding $\sigma'_{i,p} = \sigma'_i \mod p$ satisfy:

$$0 \leq \sigma'_{i,p} < \frac{p}{2^k} \tag{5.1}$$

for all $1 \leq i \leq n$, where n is the number of faulty signatures. We show how to recover p, using an attack similar to [NS98]. With LLL [LLL82], the attacker computes a short vector $(u_1, \ldots, u_n)$ such that:

$$\sum_{i=1}^{n} u_i \cdot \sigma'_i = 0 \mod N$$

This implies:

$$\sum_{i=1}^{n} u_i \cdot \sigma'_{i,p} = 0 \mod p$$

Since from (5.1) the $\sigma'_{i,p}$ are small modulo p, if the u_i's are small enough, then the equality will hold not only modulo p but also over $\mathbb{Z}$:

$$\sum_{i=1}^{n} u_i \cdot \sigma'_{i,p} = 0$$

This gives a vector $(u_1, \ldots, u_n)$ that is orthogonal in $\mathbb{Z}$ to the unknown vector $(\sigma'_{1,p}, \ldots \sigma'_{n,p})$. It is shown in [NS98] that by generating sufficiently many such vectors, one can recover the unknown vector $(\sigma'_{1,p}, \ldots \sigma'_{n,p})$ and eventually p.

Note that this attack applies to any RSA-based signature scheme with CRT, not only to PSS. This attack shows it is *not* enough for σ_p to have min-entropy, as only a few bits of entropy loss compared to the uniform distribution enable to recover p. Therefore, if we want to obtain a security proof, it seems necessary to assume that σ_p is uniformly distributed modulo p.

Actually the random fault assumption might be reasonable in practice. Namely to prevent probing attacks, the data being transmitted in the memory bus inside the micro-processor is usually encrypted. Therefore, the content of a register after a fault attack could still be some encrypted value, so it can be reasonable to model this register value as uniformly random.

5.3 PSS is Secure against Random Fault Attacks

5.3.1 The PSS Scheme

We recall the definition of the PSS scheme [BR96]. The scheme uses three hash functions $h : \{0,1\}^* \to \{0,1\}^{k_1}$, $g_1 : \{0,1\}^{k_1} \to \{0,1\}^{k_0}$ and $g_2 : \{0,1\}^{k_1} \to \{0,1\}^{k-k_0-k_1-1}$, where k, k_0 and k_1 are parameters.

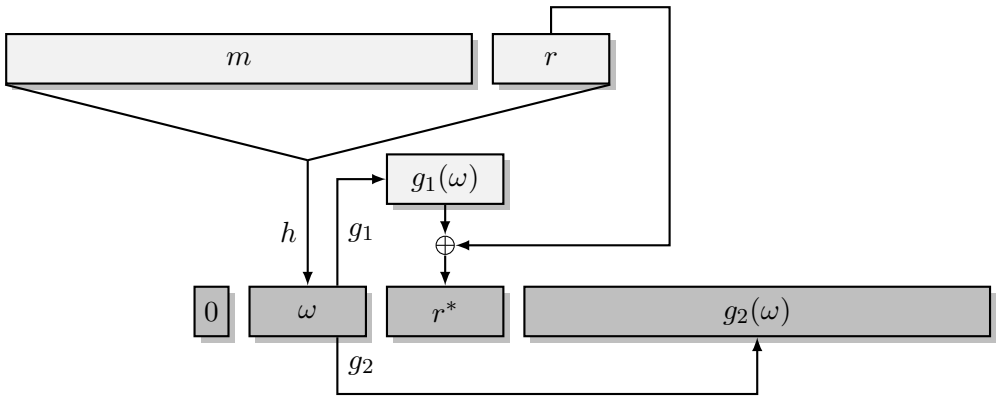

Figure 5.1: PSS: the components of the image $y = 0\|\omega\|r^*\|g_2(\omega)$ are darkened. The signature of m is $y^d \mod N$

Key Generation: generate a k-bit RSA modulus $N = pq$, and a random exponent $e \in \mathbb{Z}^*_{\phi(N)}$. Generate d such that $e \cdot d = 1 \mod \phi(N)$. The public-key is (N, e); the private key is (N, d).

Signature generation: given a message m, do the following:

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 $\sigma = y^d \mod N$

Signature Verification: given a message m and a signature σ, do the following:

1. Let $y = \sigma^e \mod N$
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.

5.3.2 Security Proof

We first give an intuition of the proof. We denote by $\mu(m, r)$ the PSS encoding scheme, that is $\mu(m, r) = 0\|\omega\|r^*\|g_2(\omega)$ where $\omega = h(m\|r)$ and $r^* = g_1(\omega) \oplus r$.

We receive as input a challenge (N, e, η) and we must output $\eta^d \mod N$. In the original PSS security proof [BR96], when receiving a signature query, the simulator generates a random α modulo N such that $\alpha^e \mod N$ can be written as $0\|\omega\|s\|t$. The simulator generates a random r of k_0 bits. Then it lets $h(m, r) = \omega$, $g_1(\omega) = s \oplus r$ and $g_2(\omega) = t$. Therefore we have that $\mu(m, r) = (\alpha^e \mod N)$. The simulator can then return α as a signature for m. When receiving a hash query for $h(m, r)$, the simulator generates a random α modulo N such that $\eta \cdot \alpha^e$ can be written as $0\|\omega\|s\|t$; it then proceeds as previously. In this case we have $\mu(m, r) = (\eta \cdot \alpha^e \mod N)$. Therefore a forgery for $\mu(m, r)$ enables to compute $\eta^d \mod N$.

One can see that if there is no collision on the randoms r used for signature generation, and no collision on the values ω, then the simulation is perfect. Then given a forgery σ' for some message m', with high probability we have that $\mu(m', r') = (\eta \cdot \alpha^e \mod N)$ for some known α. Therefore from $\sigma' = \mu(m', r')^d \mod N$ one can compute $\eta^d \mod N$ as required and solve the RSA challenge.

In our extended model of security, we must additionally simulate a faulty signature oracle. To do this, one could first generate as previously a random α modulo N such that $\alpha^e \mod N$ can be written as $0\|\omega\|s\|t$. The simulator generates a random r of k_0 bits. Then it lets $h(M, r) = \omega$, $g_1(\omega) = s \oplus r$ and $g_2(\omega) = t$, so that again $\mu(m, r) = (\alpha^e \mod N)$. Then instead of returning the correct signature α, the simulator could generate a random u modulo q, and output the faulty signature:

$$\alpha' = \alpha + u \cdot p \mod N \tag{5.2}$$

Obviously our simulator cannot do this, because it does not know the prime factors p and q. Instead we show that the distribution of α' is statistically close to uniform in $\mathbb{Z}_N$; therefore, the simulator can simply return a random $\alpha' \in \mathbb{Z}_N$.

Since RSA is a permutation, instead of considering the distribution of α', one can consider the distribution of $y' = \alpha'^e \mod N$. From (5.2) we have:

$$y' = y + v \cdot p \mod N$$

where v is uniformly distributed modulo q and y is uniformly distributed in $[0, 2^{k-1}[$. The following lemma shows that the distribution of y' is statistically close to uniform in $\mathbb{Z}_N$.

Lemma 5.1. *Let $N = pq$ be a k-bit modulus where p and q are $k/2$-bit, and let y be a random integer such that $0 \leq y < 2^{k-1}$. Let v be a random integer modulo q. Then the distribution of $y' = y + v \cdot p \mod N$ is ϵ-statistically close to uniform modulo N, with $\epsilon = \frac{4}{2^{k/2}}$*

Proof. We consider a fixed $a \in \mathbb{Z}_N$ and we provide an estimate of $\Pr[y' = a]$. For this we consider the solutions of the equation:

$$a \equiv y + v \cdot p \mod N \tag{5.3}$$

We have that for every integer $v \in [0, q)$, there exists a unique integer $y \in [0, N)$ which satisfies the above relation. However we are only interested in the y's in the range $[0, 2^{k-1})$. We have that for each $i \in [1, q]$, the pair:

$$(v = q - i,\ y = a + ip \mod N)$$

is a solution of (5.3) iff

$$a + ip \mod N < 2^{k-1} \tag{5.4}$$

Depending on the choice of a, there are actually either $\lfloor \frac{2^{k-1}}{p} \rfloor$ or $\lfloor \frac{2^{k-1}}{p} \rfloor + 1$ many i values which satisfy relation (5.4). Hence there are $\lfloor \frac{2^{k-1}}{p} \rfloor$ or $\lfloor \frac{2^{k-1}}{p} \rfloor + 1$ many solutions to congruence (5.3) such that $y < 2^{k-1}$. Since y and v are random integers in the range $[0, 2^{k-1})$ and $[0, q)$ respectively, this gives:

$$\left\lfloor \frac{2^{k-1}}{p} \right\rfloor \cdot \frac{1}{2^{k-1}} \cdot \frac{1}{q} \leq \Pr[y' = a] \leq \left(\left\lfloor \frac{2^{k-1}}{p} \right\rfloor + 1 \right) \cdot \frac{1}{2^{k-1}} \cdot \frac{1}{q}$$

We write $\lfloor \frac{2^{k-1}}{p} \rfloor = c$, which gives $p \cdot c < 2^{k-1} < p \cdot c + p$. We obtain:

$$\begin{aligned}
\Pr[y' = a] &\geq \frac{c}{2^{k-1}q} = \frac{1}{N} \cdot \frac{pc}{2^{k-1}} = \frac{1}{N} \cdot \left(1 - \frac{2^{k-1} - pc}{2^{k-1}}\right) \\
&> \frac{1}{N} \cdot \left(1 - \frac{p}{2^{k-1}}\right) \qquad (\text{as } 2^{k-1} < pc + p) \\
&> \frac{1}{N} \cdot \left(1 - \frac{2p}{N}\right) \qquad (\text{as } 2^{k-1} > \frac{N}{2}) \\
&= \left(1 - \frac{2}{q}\right) \cdot \frac{1}{N}
\end{aligned}$$

Similarly, we have:

$$\Pr[y' = a] \leq \left(1 + \frac{2}{q}\right) \cdot \frac{1}{N}$$

This gives:

$$\left(1 - \frac{2}{q}\right) \cdot \frac{1}{N} \leq \Pr[y' = a] \leq \left(1 + \frac{2}{q}\right) \cdot \frac{1}{N}$$

for all $a \in [0, N)$. This implies that the distribution of y' is $\frac{4}{2^{k/2}}$-statistically close to uniform modulo N as $q > 2^{k/2-1}$. □

Lemma 5.1 shows that it is sufficient for our simulator to return a random α' modulo N as the faulty signature. In other words, instead of first generating a random $y \in [0, 2^{k-1})$, then a random v modulo q, then $y' = y + v \cdot p$ and finally $\alpha' = y'^d \mod N$, the simulator can simply output a random α' modulo N, and such output will be statistically indistinguishable from a faulty signature.

However to this faulty signature α' corresponds a correct signature α such that:

$$\alpha = \alpha' - u \cdot p \mod N$$

where u is randomly distributed modulo q. Equivalently letting $y' = \alpha'^e \mod N$ there exists a corresponding value y with:

$$y = y' - v \cdot p \mod N \tag{5.5}$$

where v is randomly distributed modulo q such that y can be written as:

$$y = 0 \| \omega \| s \| t = \mu(m, r)$$

This implicitly defines $h(m, r) = \omega$, $g_1(\omega) = s \oplus r$ and $g_2(\omega) = t$ for the simulation of random oracles h, g_1 and g_2.

Since our simulator does not know p, it cannot compute y in equation (5.5) and therefore our simulator does not know the corresponding values of ω, s and t; therefore our simulator cannot answer the corresponding h queries, g_1 queries and g_2 queries if such queries are made by the attacker. Intuitively for h-queries it is sufficient that the set of r values is exponentially large; for this the parameter k_0 must be large enough. For g_1 and g_2 queries we must show that the adversary has a negligible probability of querying ω. This is shown in the following lemma: we show that given a faulty signature α' (or equivalently $y' = \alpha'^e \mod N$) the distribution of ω has enough variability, if the parameter k_1 is sufficiently large. This implies that ω does not need to be computed, and therefore the factorization of N is not needed for our simulation.

Lemma 5.2. *Let $N = pq$ be a k-bit modulus where p and q are $k/2$-bit, and let y be a random integer such that $0 \leq y < 2^{k-1}$. Let v be a random integer modulo q, and let $y' = y + v \cdot p \mod N$. Write $y = 0 \| \omega \| x$ where ω is k_1-bit and x is $k - k_1 - 1$ bits. Given y', for any ω' of k_1-bit we have:*

$$\Pr[\omega = \omega' | y'] \leq \frac{8}{2^{\min(k_1, k/2)}}$$

Proof. We have that:

$$\Pr[\omega = \omega' | y'] = \frac{\#(y, v) \text{ pairs, s.t. } y' = y + v \cdot p \mod N \text{ and } y = 0 \| \omega' \| x}{\#(y, v) \text{ pairs, s.t. } y' = y + v \cdot p \mod N \text{ and } 0 \leq y < 2^{k-1}}$$

For a fixed v, the value $y \mod N$ gets fixed by the relation $y' = y + v \cdot p \mod N$. Moreover at least $\lfloor \frac{q}{2} \rfloor$ of the possible v values give $y \mod N$ in the desired range between 0 and 2^{k-1}. Hence the denominator of the above fraction can be lower bounded by $\lfloor \frac{q}{2} \rfloor$.

We have that for a fixed y', the value of y is fixed modulo p; hence for a fixed ω' with $y = 0\|\omega'\|x$, the value of x is also fixed modulo p. As x is $k - k_1 - 1$-bit, over $\mathbb{Z}$ there can be at most $\lceil \frac{2^{k-k_1-1}}{p} \rceil$ many possible x values. Hence the numerator of the above fraction can be upper bounded by $\lceil \frac{2^{k-k_1-1}}{p} \rceil$.

Hence we have,

$$\Pr[\omega = \omega' | y'] \leq \frac{\lceil \frac{2^{k-k_1-1}}{p} \rceil}{\lfloor \frac{q}{2} \rfloor} < \frac{\frac{2^{k-k_1-1}}{2^{k/2-1}} + 1}{2^{k/2-2}} = \frac{2^{k-k_1-1} + 2^{k/2-1}}{2^{k-3}} < \frac{8}{2^{\min(k_1, k/2)}}$$

□

Formally, we obtain the following theorem:

Theorem 5.1. *Assume that no algorithm can invert RSA in time t' with probability better than ε'. Then the signature scheme* PSS$[k_0, k_1]$ *is $(t, q_h, q_g, q_s, q_{fs}, \varepsilon)$ secure, where*

$$\begin{aligned} t(k) &= t'(k) - [q_s(k) + q_g(k) + q_h(k) + 1] \cdot k_0 \cdot \Theta(k^3) \\ \varepsilon(k) &= \varepsilon'(k) + (q_s + q_{fs} + 1) \cdot (q_s + q_{fs} + q_h) \cdot 2^{-k_0} + 8 \cdot q_g \cdot q_{fs} \cdot 2^{-\min(k_1, k/2)} \\ &\quad + (q_h + q_s + q_{fs}) \cdot (q_h + q_g + q_s + q_{fs} + 1) \cdot 2^{-k_1} \\ &\quad + q_h \cdot q_{fs} \cdot 2^{-k_0} + 4 \cdot q_{fs} \cdot 2^{-k/2} \end{aligned}$$

Here the attacker can make at most q_h, q_g, q_s, q_{fs} number of h queries, g queries, signature queries and fault signature queries respectively.

Proof. We use a simulator which behaves in exactly same way as in original PSS security proof [BR96], in addition it answers fault queries with a uniformly random integer modulo N. Now if the attacker is successful against our simulator then we break the RSA challenge (N, e, η) as in the original paper.

We must show that any attacker which is successful against the original attack scenario will be successful against our simulator. For that, we use a sequence of games. We start with $\texttt{Game}_0$, which is exactly the attack scenario, which requires to know the factorization of N. Then we progressively modify the game, so that eventually knowledge of the factorization of N is not needed anymore. We denote by S_i the event that the attacker succeeds in $\texttt{Game}_i$.

$\texttt{Game}_0$: this is the attack scenario. We answer signature queries as specified in the signature generation algorithm, using the private exponent d. We simulate the faulty signature queries by first generating a correct signature σ and then computing $\sigma' = \sigma + u \cdot p \mod N$ for a random u modulo q. In the following for simplicity we only consider faulty signatures modulo q; faulty signatures modulo p are simulated in exactly the same way.

$\texttt{Game}_1$: we abort if there is a collision for ω at Step 2 of the signature generation algorithm, or if the random r used during signature generation has already appeared before. We call this event A_1. More precisely event A_1 happens if one of the following is true:

- The random r used in a signature oracle or faulty signature oracle query collides with either 1) the r used in a previous signature oracle or faulty signature oracle query or 2) the r used in a previous h oracle query.
- The h function output in a signature oracle or faulty signature oracle query collides with either 1) the h function outputs in previous signature oracle or faulty signature oracle queries or 2) with a previous h oracle query output or 3) a previous g oracle query input.
- The h oracle query output collides with either 1) a h function output in previous signature oracle or faulty signature oracle query or 2) a previous h oracle query output or 3) a previous g oracle query input.

We obtain:

$$\Pr[A_1] \leq (q_s + q_{fs}) \cdot (q_s + q_{fs} + q_h) \cdot 2^{-k_0} + (q_h + q_s + q_{fs}) \cdot (q_h + q_g + q_s + q_{fs}) \cdot 2^{-k_1}$$

and:

$$|\Pr[S_1] - \Pr[S_0]| \leq \Pr[A_1]$$

$\texttt{Game}_2$: we construct a similar simulator as in the original PSS security proof [BR96]; however to deal with faulty signature queries we continue to use the factorization of N.

The simulator receives as input a challenge η and must output $\eta^d \mod N$. When receiving a signature query, the simulator generates a random α modulo N such that $\alpha^e \mod N$ can be written as $0\|\omega\|s\|t$. The simulator generates a random r of k_0 bits. Then it lets $h(m, r) = \omega$, $g_1(\omega) = s \oplus r$ and $g_2(\omega) = t$.

When receiving a hash query for $h(m, r)$, the challenger generates a random α modulo N such that $\eta \cdot \alpha^e \mod N$ can be written as $0\|\omega\|s\|t$; it then defines $h(m, r) = \omega$, $g_1(\omega) = s \oplus r$ and $g_2(\omega) = t$ as previously. The queries to g_1 and g_2 are simulated by returning a random value for every new input.

To simulate the faulty signature oracle, one first generates as above a random α modulo N such that $\alpha^e \mod N$ can be written as $0\|\omega\|s\|t$. The simulator generates a random r of k_0 bits. Then it lets $h(m, r) = \omega$, $g_1(\omega) = s \oplus r$ and $g_2(\omega) = t$. Then instead of returning α, the simulator generates a random u modulo q, and outputs:

$$\alpha' = \alpha + u \cdot p \mod N \tag{5.6}$$

In $\texttt{Game}_2$ we abort as in $\texttt{Game}_1$, and additionally in the following case: while generating a random α modulo N such that $\alpha^e \mod N$ can be written as $0\|\omega\|s\|t$ during signature or faulty signature queries (and similarly for $h(m, r)$ queries), we stop after trying $k_0 + 1$ times. This adds $(q_h + q_s + q_{fs}) \cdot 2^{-k_0}$ in the error term:

$$|\Pr[S_2] - \Pr[S_1]| \leq (q_h + q_s + q_{fs}) \cdot 2^{-k_0}$$

$\texttt{Game}_3$: we abort if the attacker makes a query for $g(\omega)$ where ω was used in a faulty signature for message m and random r, while the attacker has not made a query to

$h(m, r)$ before. We define this event as A_3. As all the query answers are simulated independently, from Lemma 5.2 this gives:

$$|\Pr[S_3] - \Pr[S_2]| \leq \Pr[A_3] \leq q_g \cdot q_{fs} \cdot \frac{8}{2^{\min(k_1, k/2)}}$$

$\texttt{Game}_4$: we abort if the attacker makes a query for $h(m, r)$ where r was used to generate a faulty signature with ω, while the attacker has not made a query before to $g(\omega)$. In this case the attacker's view is independent from r, which gives:

$$|\Pr[S_4] - \Pr[S_3]| \leq q_h \cdot q_{fs} \cdot 2^{-k_0}$$

$\texttt{Game}_5$: we abort if the attacker makes a query for $h(m, r)$ where r was used to generate a faulty signature, or if the attacker makes a query for $g(\omega)$ where ω was used in a faulty signature. $\texttt{Game}_5$ is the same as $\texttt{Game}_4$ since for a faulty signature m with random r and ω, either the attacker starts with a $h(m, r)$ query or it starts with a $g(\omega)$ query.

$$\Pr[S_5] = \Pr[S_4]$$

$\texttt{Game}_6$: we change the way the faulty signature oracle is simulated. Instead of first generating α and then α' as in equation (5.6), we first generate a uniformly random α' and then a random u modulo q such that $\alpha^e \mod N$ can be written as $0\|\omega\|s\|t$. From Lemma 5.1 we have:

$$|\Pr[S_6] - \Pr[S_5]| \leq q_{fs} \cdot \frac{4}{2^{k/2}}$$

$\texttt{Game}_7$: since we do not answer the queries for $h(m, r)$ where r was used to generate a faulty signature, and the queries for $g(\omega)$ where ω was used in a faulty signature, we do not need to compute ω. Therefore, we do not need to compute a random u modulo q such that $\alpha^e \mod N$ can be written as $0\|\omega\|s\|t$. Therefore we do not need to know the factorization of N anymore, and we have:

$$\Pr[S_7] = \Pr[S_6]$$

Finally, if the adversary outputs a forgery with probability at least ε in $\texttt{Game}_0$, then the adversary must output a forgery with probability at least $\varepsilon - |\Pr[S_7] - \Pr[S_0]|$ in $\texttt{Game}_7$. As in the original PSS security proof, from this forgery we can solve the RSA challenge with probability at least:

$$\varepsilon' = \varepsilon - |\Pr[S_7] - \Pr[S_0]| - 2^{-k_1}$$

Combining the previous inequalities, we get (5.6). □

5.4 PSS-R is Secure against Fault Attacks

In PSS-R or PSS with message recovery the goal is to save bandwidth such that the message is recoverable from the signature; hence it is not necessary to send the message separately.

5.4.1 The PSS-R Scheme

We recall the definition of the PSS-R scheme [BR96]. The scheme uses three hash functions $h : \{0,1\}^* \to \{0,1\}^{k_1}$, $g_1 : \{0,1\}^{k_1} \to \{0,1\}^{k_0}$ and $g_2 : \{0,1\}^{k_1} \to \{0,1\}^{k-k_0-k_1-1}$, where k, k_0 and k_1 are the parameters.

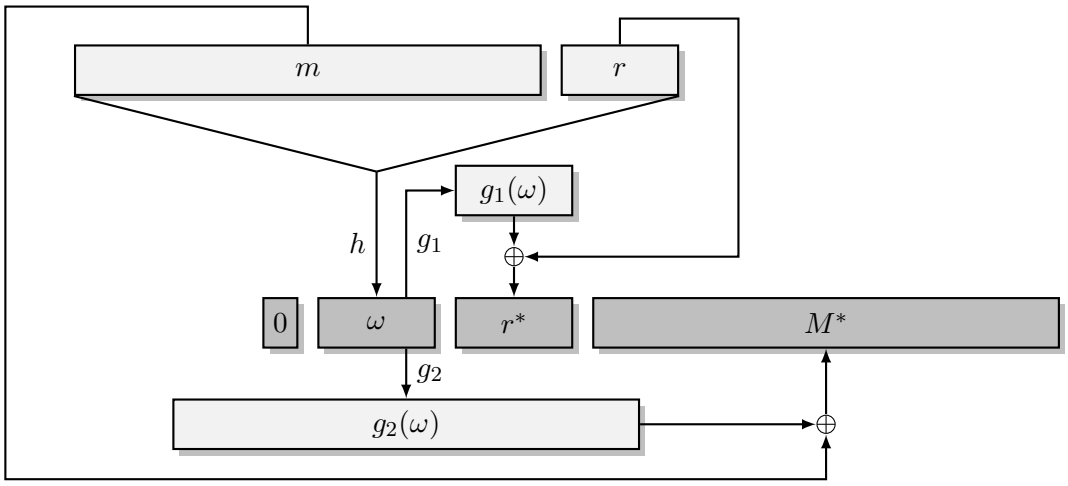

Figure 5.2: PSS-R: Components of image $y = 0\|\omega\|r^*\|M^*$ are darkened. The signature of M is $y^d \mod N$

Key Generation: generate a k-bit RSA modulus $N = pq$, and a random exponent $e \in \mathbb{Z}^*_{\phi(N)}$. Generate d such that $e \cdot d = 1 \mod \phi(N)$. The public-key is (N, e); the private key is (N, d).

Signature generation: given a message m, do the following:

1. $r \leftarrow \{0,1\}^{k_0}$
2. $\omega \leftarrow h(M\|r)$
3. $r^* \leftarrow g_1(\omega) \oplus r$
4. $m^* \leftarrow g_2(\omega) \oplus m$
5. $y \leftarrow 0\|\omega\|r^*\|m^*$
6. Return $\sigma = y^d \mod N$

Message Recovery: given a signature σ, do the following:

1. Let $y = \sigma^e \mod N$
2. Parse y as $0\|\omega\|r^*\|m^*$. If the parsing fails return REJECT.
3. $r \leftarrow r^* \oplus g_1(\omega)$
4. $m \leftarrow m^* \oplus g_2(\omega)$

5. If $h(m\|r) = \omega$ return m.

6. else return REJECT.

5.4.2 Security Proof

Theorem 5.2. *Assume that no algorithm can invert RSA in time t' with probability better than ε'. Then the signature scheme* PSS-R$[k_0, k_1]$ *is $(t, q_h, q_g, q_s, q_{fs}, \varepsilon)$ secure, where:*

$$\begin{aligned} t(k) &= t'(k) - [q_s(k) + q_g(k) + q_h(k) + 1] \cdot k_0 \cdot \Theta(k^3) \\ \varepsilon(k) &= \varepsilon'(k) + (q_s + q_{fs} + 1) \cdot (q_s + q_{fs} + q_h) \cdot 2^{-k_0} + 8 \cdot q_g \cdot q_{fs} \cdot 2^{-\min(k_1, k/2)} \\ &\quad + (q_h + q_s + q_{fs}) \cdot (q_h + q_g + q_s + q_{fs} + 1) \cdot 2^{-k_1} \\ &\quad + q_h \cdot q_{fs} \cdot 2^{-k_0} + 4 \cdot q_{fs} \cdot 2^{-k/2} \end{aligned}$$

Here the attacker can make at most q_h, q_g, q_s, q_{fs} number of h queries, g queries, signature queries and fault signature queries respectively.

Proof. The proof of this theorem is very similar to that of Theorem 5.1 and hence is omitted. □

5.5 Conclusion

We obtain from the previous theorems that unless the attacker is making more fault oracle queries than hash oracle queries, one gets the same security bound as in the original PSS proof without fault oracle. We note that in practice fault queries are usually more expensive than hash queries, since those hash queries can be made offline when a concrete hash function is used.

In [Cor02] a better security bound was given for PSS (without fault oracle). It was shown that the random size k_0 could be taken as small as $\log_2 q_s$, where q_s is the maximum number of signature queries; with $q_s = 2^{30}$ this gives $k_0 = 30$ bits. However with a fault oracle one cannot take such a small k_0, since in this case the random r could be recovered by exhaustive search and the Bellcore attack would still apply.

In summary, any parameters chosen according to the bounds in the original PSS paper [BR96] give the same level of security against fault attacks. Taking $k = 1024$, $k_0 = k_1 = 128$ as suggested by Bellare and Rogaway [BR96] would guarantee 64-bit security. However, as present day adversaries are more powerful than before, one might prefer $k = 1024$, $k_0 = k_1 = 160$ guaranteeing 80-bit security.

Chapter 6

On the Impossibility of Instantiating PSS in the Standard Model

In this chapter, we consider the problem of securely instantiating Probabilistic Signature Scheme (PSS) in the standard model. PSS, proposed by Bellare and Rogaway [BR96] is a widely deployed randomized signature scheme, provably secure (*unforgeable under adaptively chosen message attacks*) in the Random Oracle Model.

Our main result is a black-box impossibility result showing that one can not 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 *lossy trapdoor permutations* [PW08]. 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. This is a joint work with Rishiraj Bhattacharyya [BM11a].

Contents

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 [BR96] as a generic scheme based on a trapdoor permutation (like RSA). In [BR96], Bellare and Rogaway showed the scheme is secure in Random Oracle (RO) Model [BR93]. Coron improved the previous security bound in [Cor02]. In Chapter 5, we showed PSS is 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 [CGH04, DOP05, GK03, KP09], starting from famous results of Canetti et. al. [CGH04], showed that there are schemes secure in the Random Oracle model, which are uninstantiable in the standard model. Naturally, these results raise concerns about the soundness of the schemes proven secure in the 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. [DOP05] 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 [PW08]. In Crypto 2010, Kiltz et. al. [KOS10] 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 the 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 the random oracle model.

6.1.2 Overview of our Technique

We use the technique of two oracles due to Hsiao and Reyzin [HR04b] for our separation results. We construct two oracles T and G such that T implements a 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. [DOP05] 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 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) can not produce 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 [DOP05], which pre-computes 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 Previous Results

A rich body of work [GT00, GKM$^+$00, GMR01, GK03, IR89, Sim98] on blackbox separation exists in the literature starting from the seminal work of Impagliazzo and Rudich [IR89]. Regarding the separation of random oracle from the standard model, the first result was due to Canetti, Goldreich and Halevi [CGH98, CGH04] who showed an artificial albeit valid signature scheme that can not be securely instantiated by standard hash functions. Many such results [DOP05, FS10, GK03, KP09] were subsequently published. To obtain our separation results we use the two oracle technique of Hsiao and Reyzin[HR04b]. The most relevant results to our work is the work of Dodis et. al. [DOP05] and of Kiltz and Pietrzak [KP09]. In [DOP05], Dodis, Oliviera and Pietrzak showed that the popular Full Domain Hash (FDH) signature scheme can not be instantiated (using blackbox technique) in the standard model by a ideal trapdoor permutation. Kiltz and Pietrzak [KP09] established that there is no blackbox reduction of any padding based CCA secure encryption scheme from ideal trapdoor permutations. In [Pai07], Paillier showed

impossibility of reduction of many RSA based signatures including PSS from different security assumptions of RSA. However, their result is based on an additional assumption (namely, instance non-malleability) of RSA. In comparison, our result is more generic as we rule out blackbox reduction from any property of random permutation.

Differences from Dodis et al's Crypto'05 paper [DOP05]

Although our definition of oracles are quite similar to that in [DOP05], difference comes in when finally implementing a forgery. The technique of [DOP05] 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 [DOP05], 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.

6.2 Preliminaries

6.2.1 Notations

Throughout the section, if x is a string, $|x|$ denotes the length of the string. 1^n denotes the string of n many 1s. If S is a set $|S|$ denotes the cardinality of the set. We use $negl(n)$ to denote any function $\gamma : \mathbb{N} \rightarrow [0,1]$ where for any constant $c > 0$ there exist 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 $n > n_0$, $f(n) > n^c$.

6.2.2 Trapdoor Permutations (TDPs)

Definition 6.1. *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* f_{pk} *over* $\{0,1\}^n$ *and* $F^{-1}(td, .)$ *implements the corresponding inverse* f_{pk}^{-1}.

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 notion for Trapdoor Permutations are known. Like [DOP05, KP09], we consider a wide class of security properties using the notion of *δ-hard games*.

6.2.3 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 [DOP05], by $\langle C, A\rangle = 1$.

Definition 6.2. *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*

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

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 [KP09] as a generalization of hard games considered in [DOP05]. It was pointed out in [KP09] that the result of [DOP05] can easily be extended to this notion.

6.2.4 Ideal Trapdoor Permutations

The notion of Ideal Trapdoor permutation was coined in [KP09]. 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, $\mathrm{Adv}_C(A, n) - \delta(C)$ 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 6.3. *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 (see [DOP05] for proof). However as we are proving negative result, showing that PSS cannot be reduced to ideal trapdoor permutation (hence to any hard game) makes our result stronger. This implies that PSS cannot be black-box reduced to security notions like collision resistant hashing, pseudo-random functions, IND-CCA secure public key encryption schemes etc.

6.2.5 Lossy Trapdoor Permutations(LTDPs)

Lossy Trapdoor Functions were introduced by Peikert et. al. in [PW08]. In this work 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 (f, f^{-1}) where f is a permutation over $\{0,1\}^n$ and f^{-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

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

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

6.3 Signature Schemes

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

- The *key generation algorithm* `Gen` is a probabilistic algorithm which given 1^k, outputs a pair of matching public and private keys, (pk, td).

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

- The *verification algorithm* `Verify` takes a message M, a candidate signature σ' and pk. It returns a bit $\mathtt{Verify}_{pk}(M, \sigma')$, equal to one if the signature is accepted, and zero otherwise. We require that if $\sigma \leftarrow \mathtt{Sign}_{td}(M)$, then $\mathtt{Verify}_{pk}(M, \sigma) = 1$.

6.3.1 Security of a Signature Scheme

In the *existential unforgeability under an adaptive chosen message attack* scenario, the forger can dynamically obtain signatures of messages of his choice and attempts to output a valid forgery. A *valid forgery* is a message/signature pair (M, x) such that $\mathtt{Verify}_{pk}(M, x) = 1$ whereas the signature of M was never requested by the forger.

6.3.2 Probabilistic Signature Scheme(PSS)

Let $TDP = (Tdg, F, F^{-1})$ be a trapdoor permutation. 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 $g_2 : \{0,1\}^{k_1} \to \{0,1\}^{k-k_0-k_1-1}$, where k, k_0 and k_1 are parameters.

$\mathtt{Gen}(1^k)$

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

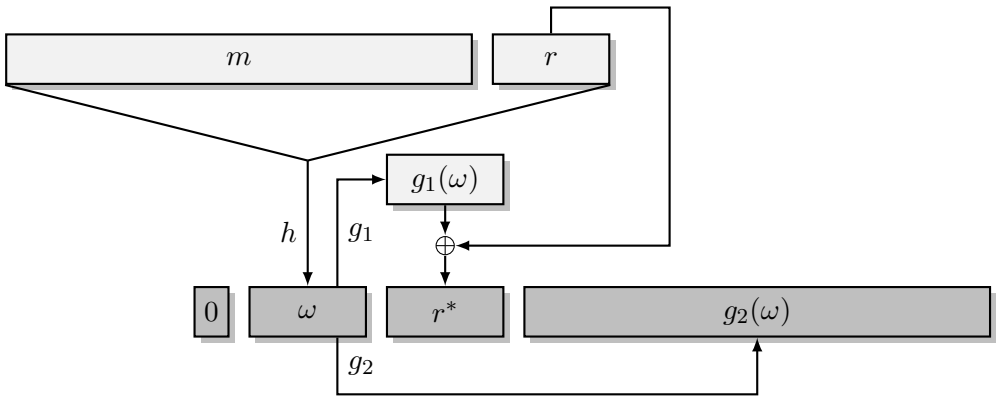

Figure 6.1: PSS_H^{TDP}: The components of the image $y = 0\|\omega\|r^*\|g_2(\omega)$ are darkened. The signature of m is $F^{-1}(td, y)$

$\texttt{Sign}_{td}(m)$

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 $\sigma = F^{-1}(td, y)$.

$\texttt{Verify}_{pk}(m, \sigma)$

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. 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

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

$PSS_H^{f_{pk}}(m\|r)$ 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 f_{pk}.* For the rest of this section, PSS_H^{TDP} denotes the signature scheme, where as $PSS_H^{f_{pk}}(\cdot)$ 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 1 A collision after the PSS transformation implies collision in the

random space. In other words,

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

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 $PSS_H^{f_{pk}}$ and hence the signatures are always different (irrespective of whether the messages are same or not)! As a side note, all our results are valid not only for PSS, but for any randomized signature scheme where the randomness is recoverable and hence the Observation 1 holds true.

6.4 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 the 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 the 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 $PSS_H^{f_{pk}}(0\|r_1) \neq PSS_H^{f_{pk}}(0\|r_2)$. So the set $\{PSS_H^{f_{pk}}(0\|r)|r \in \{0,1\}^{k_0}\}$ is of super-polynomial size. Even if G returns one random signature (from a choice of super-polynomially 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 [HR04b], 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 $f_0, f_1, f_2, \cdots, f_{2^n-1}$ 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)
- $T_2(pk, y) \to f_{pk}(y)$ (evaluate)
- $T_3(td, z) \to f^{-1}_{g(td)}(z)$ (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 $TDP^T = (Tdg, F_T, F_T^{-1})$.

- $Tdg(1^n)$ 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 $f^{-1}_{pk}(PSS^{f_{pk}}_H(0\|r))$. Here pk is the public key generated by $Tdg(1^k)$.

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

Lemma 6.1. *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 $f^{-1}_{pk}(PSS^{f_{pk}}_H(.))$ for a H of our choice! But we note that $G(.)$ samples one z' from a set of super-polynomial size and outputs $f^{-1}_{pk}(z)$. Even if the adversary sets $PSS^{f_{pk}}_H(0, r)$ for one r to be the challenge z she received, probability that $f_{pk}(G(.)) = z$ is negligible. On the other hand if $f_{pk}(G(.)) \neq z$, then knowledge of inverse of some other point does not help the adversary to find $f^{-1}_{pk}(z)$ with significant probability for a pseudorandom f_{pk}. Following the above discussion we have Lemma 6.2,

Lemma 6.2. *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 $c > 0$. Now We shall 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 it must have queried it.

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 $1/2^k$; 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 $\{0,1\}^n \setminus L_y$. 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 $1/(2^k - |L|)$. 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 $\{0,1\}^n \setminus L_x$, 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 $\{0,1\}^n \setminus L_y$. 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 6.3.

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

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 $PSS_H^{f_{pk}}$, there can be at most one $r \in \{0,1\}^{k_0}$ such that $PSS_H^{f_{pk}}(0||r) = z$. 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, 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 6.1 and Lemma 6.3, we get the main result of this section as follows

Theorem 6.1. *There is no blackbox reduction of Security under no message attack of Probabilistic Signature Scheme with super-polynomial randomness space from Oneway Trapdoor Permutations.*

6.5 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 6.2. *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 exist 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 6.4.

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,
$$PSS_H^{f_{pk}}(m_1\|r_1) = f_{pk}(\sigma_1), \cdots, PSS_H^{f_{pk}}(m_t\|r_t) = f_{pk}(\sigma_t).$$
2. $\sigma_i \neq \sigma_j$ (or equivalently $PSS_H^{f_{pk}}(m_i\|r_i) \neq PSS_H^{f_{pk}}(m_j\|r_j)$) for all $1 \leq i < j \leq t$.
3. $\{PSS_H^{f_{pk}}(m_1\|r_1), \cdots, PSS_H^{f_{pk}}(m_t\|r_t)\} \cap Y_{f_{pk}}^{PSS_H}(r_1, \cdots, r_t) = \emptyset$ where
$$Y_{f_{pk}}^{PSS_H}(r_1, \cdots, r_t) = \{f_{pk}(x) | \exists i,\ 1 \leq i \leq t, \\ PSS_H^{f_{pk}}(m_i\|r_i) \textit{ makes the oracle query } x\}.$$

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

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

Lemma 6.4. *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 it's 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 pre-computes $PSS_H^{f_{pk}}(m_i\|r)$ for all $r \in \{0,1\}^{k_0}$ and $i = 1 \cdots t$ and checks whether the Condition 3 from Section 6.5 would get satisfied or not for any possible choice of r by the `Sign` oracle. If not B can find some m_i, m_j, r_i, r_j, x such that
$$PSS_H^{f_{pk}}(m_i\|r_i) = f_{pk}(x),$$
where $PSS_H^{f_{pk}}(m_j\|r_j)$ makes the oracle query $f_{pk}(x)$. 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
$$PSS_H^{f_{pk}}(m_i\|r_i) = PSS_H^{f_{pk}}(m_j\|r_i)$$
for some $i < j \leq t$. Because of Observation 1 it is sufficient to check with the fixed r_i for collision detection purposes. If the above condition gets true again B^G can

Algorithm 2 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: $V = \{PSS_H^{f_{pk}}(m_i \| r_i^k) : 1 \leq i \leq t, 1 \leq k \leq t\}$
3: $Y = \{f_{pk}(x) |\ \exists i, k,\ 1 \leq i \leq t,\ 1 \leq k \leq t$
4: $\qquad$ *s. t.* $PSS_H^{f_{pk}}(m_i, r_i^k)$ *makes oracle query* $f_{pk}(x)\}$
5: **if** $V \cap Y \neq \emptyset$ **then**
6: $\quad$ Output Direct Forgery
7: **end if**

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 6.5 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 super-polynomial size, hence B^G cannot pre-compute all the possible outputs of $PSS_H^{f_{pk}}(m\|\cdot)$ 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).

Success Probability of B^G in Case II:

In Line 3 of Algorithm 3, Condition 1 and Condition 2 are always satisfied. So B^G can abort only in two ways.

1. In Line 3 of Algorithm 3, Σ_i becomes empty for some i, $1 \leq i \leq t$.
2. In Line 3 of Algorithm 3, 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 exists some i, j, $1 \leq i, j \leq t$, $i \neq j$ such that

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

where $PSS_H^{f_{pk}}(m_j \| r_j)$ makes the oracle query x.

Moreover, in both the cases no forgery was found in Algorithm 2.

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

Algorithm 3 B^G : Phase-II

for $i = 1$ to t **do**
  $\sigma_i^1 \leftarrow \texttt{Sign}(m_i), \cdots, \sigma_i^t \leftarrow \texttt{Sign}(m_i)$
  $\Sigma_i = \{\sigma_i^1, \cdots, \sigma_i^t\}$
  Recover $r_i^1, \cdots, r_i^t$ from $\sigma_i^1, \cdots \sigma_i^t$ using `Verify`.
  for $j = i + 1$ to t **do**
    if $PSS_H^{f_{pk}}(m_i \| r_i^k) == PSS_H^{f_{pk}}(m_j \| r_i^k)$ *for some* $1 \leq k \leq t$ **then**
      Output Direct Forgery (m_j, σ_i^k)
    end if
  end for
  for $k = 1$ to t **do**
    $X_{i,k} \leftarrow \{x | PSS_H^{f_{pk}}(m_i \| r_i^k)$ *makes oracle query* $f_{pk}(x)\}$
    if $\sigma_i^k \in X_{i,k}$ **then**
      $\Sigma_i \leftarrow \Sigma_i \setminus \{\sigma_i^k\}$
    end if
  end for
  if $\Sigma_i = \emptyset$ **then**
    Output $\perp$
  end if
  Pick any $\sigma_i \in \Sigma_i$
end for
if $\sigma_1, \cdots, \sigma_t$ satisfy Condition 1, Condition 2 and Condition 3 from Section 6.5 **then**
  Output forgery via G
else
  Output $\perp$
end if

the set of r for which $PSS_H^{f_{pk}}(m_i, r) = f_{pk}(x)$ and x was queried while computing $PSS_H^{f_{pk}}(m_i, r)$ 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 $r_i^1, r_i^2, \cdots, r_i^t$ was 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 f_{pk} queries (made by $PSS_H^{f_{pk}}$) 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 f_{pk} queries made while computing Σ_j is at most $t|H|$. So overall, for all $j \neq i$, total number of f_{pk} queries made by the $PSS_H^{f_{pk}}$ 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 f_{pk} queries is at most $\frac{t^4}{2^{k_0}}$.

Hence we get that

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

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

□

G does not break the security of TDP^T

Lemma 6.5. *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 [DOP05], where one argues in the absence of oracle G the claim holds because of computational indistinguishability of f_{pk} from a random permutation. Moreover, Lemma 6.6 below states the accepting condition of oracle G can only be satisfied with a negligible probability. □

Lemma 6.6. *Let f 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 f)*

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

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

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

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

Lemma 6.6 can proved following the same technique of Lemma 3 of [DOP05]. For completeness we give the following proof.

Proof. Consider any oracle TM A, where A^f 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 f, then the probability that the relation $f^{-1}(PSS_H^f(m_i\|r_i)) = x_i$ holds for some r_i is negligible. Let X_f^A, $|X_f^A| = n^c$ denote all the oracle queries made by A^f, i.e.

$$X_f^A = \{x | A^f \textit{ makes the oracle query } x\}.$$

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 $X_f^h(r_1, \cdots, r_t) = \{f^{-1}(y) | y \in Y_f^h(r_1, \cdots, r_t)\}$, i.e.

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

and let

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

Condition 3 states that $f(H(r_1, \cdots, r_t)) \cap f(X_f^h(r_1, \cdots, r_t)) = \phi$, and as f is permutation this is equivalent to

$$H(r_1, \cdots, r_t) \cap X_f^h(r_1, \cdots, r_t) = \phi.$$

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

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

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^f 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 $\texttt{negl}(n)$. □

[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 $2^{n - c\log n - n} = \frac{1}{n^c}$

6.6 No Reduction from Lossy Trapdoor Permutations

Lossy Trapdoor Functions, introduced by Peikert et. al. has gained considerable attention in recent years. In a recent work [KOS10], has proven IND-CPA security of OAEP under Lossy Trapdoor Permutation. Moreover different constructions like IND-CCA secure encryption, which cannot be reduced to standard trapdoor permutation using blackbox techniques, were proven reducible to 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 as well. Specifically, Let $LTDP = (S, F, F')$ be a family of Lossy Trapdoor Permutation. We define the output of PSS based on LTDP as $\sigma = f^{-1}(PSS_H(m||r))$ where $(f, f^{-1}) \in F$. Note that, while instantiating PSS by a lossy TDP, we consider the trapdoor permutation to be the injective mode of the TDP.

Theorem 6.3. *There is no blackbox reduction of existential unforgeability against chosen message attack of Probabilistic Signature Scheme from Lossy Trapdoor Permutations.*

Proof To prove Theorem 6.3, we need new definitions of the oracles.

Definition of $\mathcal{T}$

$\mathcal{T}$ is defined as a pair (T, T'). Choose $2^n + 1$ permutations $f_0, \cdots, f_{2^n-1}$ 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:

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

On the other hand T' is defined as follows

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

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

- $S(b)$ If $b = 1$, choose a uniform random $td \leftarrow \{0,1\}^n$ computed $pk = T_1(td)$ 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 6.7. *$LTDP^{T,T'}$ 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 form a random function (permutation), the lemma follows. □

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 $\{0,1\}^* \backslash 0$ 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.

- $F(pk, \sigma_i) = PSS_H^{f_{pk}}(m_i \| r)$ 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

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

Finally G checks whether F is the lossy mode[2], if yes it aborts; otherwise G chooses one r uniformly at random from $\{0,1\}^{k_0}$ and computes the PSS hash of

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

$0\|r$ 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 6.6, it happens with negligible probability and we get the following result.

Lemma 6.8. *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 6.4. This completes the proof of Theorem 6.3. □

6.7 No Reduction from Hard Games with Inversion

Like [DOP05], 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 [DOP05], if we modify G to ask for signatures of $|H| + q(n)$ messages and modify Lemma 6.6 accordingly, we get the following two theorems.

Theorem 6.4. *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. *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.*

6.8 Conclusion

Following the negative results, on generic insecurity of FDH by Dodis et. al [DOP05] and of OAEP by Kiltz and Pietrzak [KP09] in the standard model, we show security of PSS also can not be black box reduced to any property of an ideal trapdoor permutation. Moreover, we also show one can not even hope to achieve security of PSS based on Lossy Trapdoor Permutations. On the contrary recently a secure instantiation of OAEP has been realized based on Lossy Trapdoor Permutations [KOS10].

Bibliography

[ADR02] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In *EUROCRYPT*, pages 83–107, 2002.

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT*, pages 223–238, 2004. 18

[BBN+09] Mihir Bellare, Zvika Brakerski, Moni Naor, Thomas Ristenpart, Gil Segev, Hovav Shacham, and Scott Yilek. Hedged public-key encryption: How to protect against bad randomness. In *ASIACRYPT*, pages 232–249, 2009.

[BCFW09] Alexandra Boldyreva, David Cash, Marc Fischlin, and Bogdan Warinschi. Foundations of non-malleable hash and one-way functions. In *ASIACRYPT*, pages 524–541, 2009.

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In *EUROCRYPT*, pages 37–51, 1997. 92

[BDL01] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of eliminating errors in cryptographic computations. *J. Cryptology*, 14(2):101–119, 2001. 92

[BDPA08] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In *EUROCRYPT*, pages 181–197, 2008.

[BK09] Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full aes-192 and aes-256. In *ASIACRYPT*, pages 1–18, 2009. 23

[BKN09] Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolic. Distinguisher and related-key attack on the full aes-256. In *CRYPTO*, pages 231–249, 2009. 23

[Bla06] John Black. The ideal-cipher model, revisited: An uninstantiable blockcipher-based hash function. In *FSE*, pages 328–340, 2006. 22

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *ASIACRYPT*, pages 514–532, 2001. 18

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *J. Cryptology*, 17(4):297–319, 2004. 18

[BM11a] Rishiraj Bhattacharyya and Avradip Mandal. On the impossibility of instantiating pss in the standard model. In *Public Key Cryptography*, pages 351–368, 2011. 9, 105

[BM11b] Rishiraj Bhattacharyya and Avradip Mandal. On the indifferentiability of fugue and luffa. In *ACNS*, pages 479–497, 2011.

[BMN09] 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.

[BMN10] Rishiraj Bhattacharyya, Avradip Mandal, and Mridul Nandi. Security analysis of the mode of jh hash function. In *FSE*, pages 168–191, 2010.

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In *EUROCRYPT*, pages 139–155, 2000.

[BR93] 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. 6, 8, 18, 25, 55, 95, 106

[BR94] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In *EUROCRYPT*, pages 92–111, 1994. 6

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures - how to sign with rsa and rabin. In *EUROCRYPT*, pages 399–416, 1996. 6, 8, 9, 18, 55, 92, 93, 96, 97, 100, 101, 103, 104, 105, 106

[BR02] John Black and Phillip Rogaway. A block-cipher mode of operation for parallelizable message authentication. In *EUROCRYPT*, pages 384–397, 2002.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *EUROCRYPT*, pages 409–426, 2006. 57, 58, 69

[BRS02] 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, 55

[CDMP05] 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. 7, 8, 13, 15, 16, 19, 21, 22, 23, 36, 37, 52, 54, 58, 63

[CDMS10] Jean-Sébastien Coron, Yevgeniy Dodis, Avradip Mandal, and Yannick Seurin. A domain extender for the ideal cipher. In *TCC*, pages 273–289, 2010. 8, 21, 56

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *STOC*, pages 209–218, 1998. 6, 9, 53, 55, 57, 73, 108

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004. 6, 9, 12, 95, 106, 108

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT*, pages 255–271, 2003. 18

[CHK07] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. *J. Cryptology*, 20(3):265–294, 2007. 18

[CJK+09] Jean-Sébastien Coron, Antoine Joux, Ilya Kizhvatov, David Naccache, and Pascal Paillier. Fault attacks on rsa signatures with partially unknown messages. In *CHES*, pages 444–456, 2009. 92, 93

[CJM+11] Jean-Sébastien Coron, Antoine Joux, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Cryptanalysis of the rsa subgroup assumption from tcc 2005. In *Public Key Cryptography*, pages 147–155, 2011.

[CLNY06] 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.

[CM09] Jean-Sébastien Coron and Avradip Mandal. Pss is secure against random fault attacks. In *ASIACRYPT*, pages 653–666, 2009. 9, 91

[CMNT11] Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In *CRYPTO*, pages 487–504, 2011.

[CMPP05] Benoît Chevallier-Mames, Duong Hieu Phan, and David Pointcheval. Optimal asymmetric encryption and signature paddings. In *ACNS*, pages 254–268, 2005. 56

[CN08] Donghoon Chang and Mridul Nandi. Improved indifferentiability security analysis of chopmd hash function. In *FSE*, pages 429–443, 2008.

[Cop97] Don Coppersmith. Small solutions to polynomial equations, and low exponent rsa vulnerabilities. *J. Cryptology*, 10(4):233–260, 1997. 93

[Cor02] Jean-Sébastien Coron. Optimal security proofs for pss and other signature schemes. In *EUROCRYPT*, pages 272–287, 2002. 18, 55, 92, 104, 106

[CPS08] 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. 19, 52, 55, 56, 65, 66, 70

[CS06a] Debrup Chakraborty and Palash Sarkar. Hch: A new tweakable enciphering scheme using the hash-encrypt-hash approach. In *INDOCRYPT*, pages 287–302, 2006. 22, 28

[CS06b] Debrup Chakraborty and Palash Sarkar. A new mode of encryption providing a tweakable strong pseudo-random permutation. In *FSE*, pages 293–309, 2006. 22, 28

[CS08] Debrup Chakraborty and Palash Sarkar. Hch: A new tweakable enciphering scheme using the hash-counter-hash approach. *IEEE Transactions on Information Theory*, 54(4):1683–1699, 2008. 22, 28

[Dam89] Ivan Damgård. A design principle for hash functions. In *CRYPTO*, pages 416–427, 1989. 54

[Des00] Anand Desai. The security of all-or-nothing encryption: Protecting against exhaustive key search. In *CRYPTO*, pages 359–375, 2000. 22

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. 2

[DOP05] Yevgeniy Dodis, Roberto Oliveira, and Krzysztof Pietrzak. On the generic insecurity of the full domain hash. In *CRYPTO*, pages 449–466, 2005. 8, 106, 107, 108, 109, 110, 120, 121, 124

[DP06] Yevgeniy Dodis and Prashant Puniya. On the relation between the ideal cipher and the random oracle models. In *TCC*, pages 184–206, 2006. 8, 16, 17, 23, 37, 40, 55

[DP07] Yevgeniy Dodis and Prashant Puniya. Feistel networks made public, and applications. In *EUROCRYPT*, pages 534–554, 2007. 54

[DRRS09] 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.

[DRS09] Yevgeniy Dodis, Thomas Ristenpart, and Thomas Shrimpton. Salvaging merkle-damgård for practical applications. In *EUROCRYPT*, pages 371–388, 2009. 8, 16, 18, 19, 53, 55, 56, 60, 63, 73

[EM91] Shimon Even and Yishay Mansour. A construction of a cipher from a single pseudorandom permutation. In *ASIACRYPT*, pages 210–224, 1991. 22

[EM97] Shimon Even and Yishay Mansour. A construction of a cipher from a single pseudorandom permutation. *J. Cryptology*, 10(3):151–162, 1997. 22

[EMV08] EMV. Integrated circuit card specifications for payment systems, book 2. security and key management. Technical Report 4.2, EMVco, June 2008. `www.emvco.com`. 93

[FLP08] Marc Fischlin, Anja Lehmann, and Krzysztof Pietrzak. Robust multi-property combiners for hash functions revisited. In *ICALP (2)*, pages 655–666, 2008.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986. 6, 18

[FS10] Marc Fischlin and Dominique Schröder. On the impossibility of three-move blind signature schemes. In *EUROCRYPT*, pages 197–215, 2010. 108

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[GH11] Craig Gentry and Shai Halevi. Implementing gentry's fully-homomorphic encryption scheme. In *EUROCRYPT*, pages 129–148, 2011.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the fiat-shamir paradigm. In *FOCS*, pages 102–, 2003. 106, 108

[GKM+00] 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. 108

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[GMR01] Yael Gertner, Tal Malkin, and Omer Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *FOCS*, pages 126–135, 2001. 108

[GR04] Craig Gentry and Zulfikar Ramzan. Eliminating random permutation oracles in the even-mansour cipher. In *ASIACRYPT*, pages 32–47, 2004. 76

[Gra02] Louis Granboulan. Short signatures in the random oracle model. In *ASIACRYPT*, pages 364–378, 2002. 22, 37, 56

[Gro05] Jens Groth. Cryptography in subgroups of z_n. In *TCC*, pages 50–65, 2005.

[GT00] Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *FOCS*, pages 305–313, 2000. 108

[Hal07] Shai Halevi. Invertible universal hashing and the tet encryption mode. In *CRYPTO*, pages 412–429, 2007. 22, 28

[HKT11] Thomas Holenstein, Robin Künzler, and Stefano Tessaro. The equivalence of the random oracle model and the ideal cipher model, revisited. In *STOC*, pages 89–98, 2011. 19, 21, 22, 23, 34, 37, 55, 56, 75, 89

[HPY07] 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.

[HR03] Shai Halevi and Phillip Rogaway. A tweakable enciphering mode. In *CRYPTO*, pages 482–499, 2003. 27, 28

[HR04a] Shai Halevi and Phillip Rogaway. A parallelizable enciphering mode. In *CT-RSA*, pages 292–304, 2004. 27, 28, 29

[HR04b] 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. 8, 107, 108, 113

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *STOC*, pages 44–61, 1989. 108

[ISO02] ISO/IEC 9796-2. Information technology – security techniques – digital signature schemes giving message recovery – part 2: Integer factorization based mechanisms. Technical report, ISO, 2002. 92

[Jon02] Jakob Jonsson. An oaep variant with a tight security proof. *IACR Cryptology ePrint Archive*, 2002:34, 2002. 22

[KÖ9] Robin Künzler. Are the random oracle and the ideal cipher models equivalent? Master's thesis, ETH Zurich, Switzerland, 2009. 55

[KOS10] Eike Kiltz, Adam O'Neill, and Adam Smith. Instantiability of rsa-oaep under chosen-plaintext attack. In *CRYPTO*, pages 295–313, 2010. 107, 122, 124

[KP09] 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. 8, 10, 106, 108, 110, 124

[KR01] Joe Kilian and Phillip Rogaway. How to protect des against exhaustive key search (an analysis of desx). *J. Cryptology*, 14(1):17–35, 2001. 22

[KR07] Lars R. Knudsen and Vincent Rijmen. Known-key distinguishers for some block ciphers. In *ASIACRYPT*, pages 315–324, 2007. 56, 75, 76

[Kro06] Ted Krovetz. Message authentication on 64-bit architectures. In *Selected Areas in Cryptography*, pages 327–341, 2006. 25

[KW03] Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In *ACM Conference on Computer and Communications Security*, pages 155–164, 2003. 56

[LLL82] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982. 93, 95

[LR88] Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.*, 17(2):373–386, 1988. 23, 54

[LRW02] Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. In *CRYPTO*, pages 31–46, 2002. 24, 40

[LRW11] Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. *J. Cryptology*, 24(3):588–613, 2011. 24, 40

[Man07] Avradip Mandal. Mac constructions: Security bounds and distinguishing attacks. Master's thesis, University of Waterloo, Canada, 2007.

[Mau92] Ueli M. Maurer. A simplified and generalized treatment of luby-rackoff pseudorandom permutation generator. In *EUROCRYPT*, pages 239–255, 1992. 54

[Mau02] Ueli M. Maurer. Indistinguishability of random systems. In *EUROCRYPT*, pages 110–132, 2002. 58

[Mer89] Ralph C. Merkle. One way hash functions and des. In *CRYPTO*, pages 428–446, 1989. 54

[MF07] David A. McGrew and Scott R. Fluhrer. The security of the extended codebook (xcb) mode of operation. In *Selected Areas in Cryptography*, pages 311–327, 2007. 22, 28

[Min09] Kazuhiko Minematsu. Beyond-birthday-bound security based on tweakable block cipher. In *FSE*, pages 308–326, 2009. 24, 25

[MM07] Kazuhiko Minematsu and Toshiyasu Matsushima. New bounds for pmac, tmac, and xcbc. In *FSE*, pages 434–451, 2007.

[MPN10] Avradip Mandal, Jacques Patarin, and Valérie Nachef. Indifferentiability beyond the birthday bound for the xor of two public random permutations. In *INDOCRYPT*, pages 69–81, 2010.

[MPS12] Avradip Mandal, Jacques Patarin, and Yannick Seurin. On the public indifferentiability and correlation intractability of the 6-round feistel construction. In *TCC*, pages 285–302, 2012. 9, 53

[MRH04] 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. 7, 8, 11, 12, 13, 14, 15, 16, 18, 22, 25, 58, 60

[MT07] Ueli M. Maurer and Stefano Tessaro. Domain extension of public random functions: Beyond the birthday barrier. In *CRYPTO*, pages 187–204, 2007.

[NM08] Mridul Nandi and Avradip Mandal. Improved security analysis of PMAC. *J. Math. Cryptol.*, 2(2):149–162, 2008.

[NR99] Moni Naor and Omer Reingold. On the construction of pseudorandom permutations: Luby-rackoff revisited. *J. Cryptology*, 12(1):29–66, 1999. 54

[NS98] Phong Q. Nguyen and Jacques Stern. Cryptanalysis of a fast public key cryptosystem presented at sac '97. In *Selected Areas in Cryptography*, pages 213–218, 1998. 95

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC*, pages 427–437, 1990.

[P1304] IEEE P1363a. Standard specifications for public key cryptography: Additional techniques. Technical report, IEEE, 2004. available at `http://www.manta.ieee.org/groups/1363`.

[Pai07] Pascal Paillier. Impossibility proofs for rsa signatures in the standard model. In *CT-RSA*, pages 31–48, 2007. 108

[Pat90] Jacques Patarin. Pseudorandom permutations based on the des scheme. In *EUROCODE*, pages 193–204, 1990. 54

[Pat91] Jacques Patarin. New results on pseudorandom permutation generators based on the des scheme. In *CRYPTO*, pages 301–312, 1991. 54

[Pat98] Jacques Patarin. About feistel schemes with six (or more) rounds. In *FSE*, pages 103–121, 1998. 54

[Pat03] Jacques Patarin. Luby-rackoff: 7 rounds are enough for $2^{n(1\text{-epsilon})}$security. In *CRYPTO*, pages 513–529, 2003. 54

[Pat04] Jacques Patarin. Security of random feistel schemes with 5 or more rounds. In *CRYPTO*, pages 106–122, 2004. 54

[PGV93] Bart Preneel, René Govaerts, and Joos Vandewalle. Hash functions based on block ciphers: A synthetic approach. In *CRYPTO*, pages 368–378, 1993. 55

[PP03] Duong Hieu Phan and David Pointcheval. Chosen-ciphertext security without redundancy. In *ASIACRYPT*, pages 1–18, 2003. 22, 23, 37

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *STOC*, pages 187–196, 2008. 10, 105, 107, 111

[RBB03] Phillip Rogaway, Mihir Bellare, and John Black. Ocb: A block-cipher mode of operation for efficient authenticated encryption. *ACM Trans. Inf. Syst. Secur.*, 6(3):365–403, 2003. 22

[RR00] Zulfikar Ramzan and Leonid Reyzin. On the round security of symmetric-key cryptographic primitives. In *CRYPTO*, pages 376–393, 2000. 54

[RSA78] 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. 92

[RSS11] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In *EUROCRYPT*, pages 487–506, 2011. 12, 14

[Seu09] Yannick Seurin. *Primitives et protocoles cryptographiques à sécurité prouvée*. PhD thesis, Université de Versailles Saint-Quentin-en-Yvelines, France, 2009. 55, 70

[Sho04] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, 2004:332, 2004. 11, 42

[Sim98] 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

[Vau03] Serge Vaudenay. Decorrelation: A theory for block cipher security. *J. Cryptology*, 16(4):249–286, 2003. 54

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *EUROCRYPT*, pages 24–43, 2010.

[WFW05] Peng Wang, Dengguo Feng, and Wenling Wu. Hctr: A variable-input-length enciphering mode. In *CISC*, pages 175–188, 2005. 22, 28

[YMO09] Kazuki Yoneyama, Satoshi Miyagawa, and Kazuo Ohta. Leaky random oracle. *IEICE Transactions*, 92-A(8):1795–1807, 2009. 18, 53, 55, 56, 60

List Of Publications

[BM11a] Rishiraj Bhattacharyya and Avradip Mandal. On the impossibility of instantiating pss in the standard model. In *Public Key Cryptography*, pages 351–368, 2011. 9, 105

[BM11b] Rishiraj Bhattacharyya and Avradip Mandal. On the indifferentiability of fugue and luffa. In *ACNS*, pages 479–497, 2011.

[BMN09] 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.

[BMN10] Rishiraj Bhattacharyya, Avradip Mandal, and Mridul Nandi. Security analysis of the mode of jh hash function. In *FSE*, pages 168–191, 2010.

[CDMS10] Jean-Sébastien Coron, Yevgeniy Dodis, Avradip Mandal, and Yannick Seurin. A domain extender for the ideal cipher. In *TCC*, pages 273–289, 2010. 8, 21, 56

[CJM+11] Jean-Sébastien Coron, Antoine Joux, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Cryptanalysis of the rsa subgroup assumption from tcc 2005. In *Public Key Cryptography*, pages 147–155, 2011.

[CM09] Jean-Sébastien Coron and Avradip Mandal. Pss is secure against random fault attacks. In *ASIACRYPT*, pages 653–666, 2009. 9, 91

[CMNT11] Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In *CRYPTO*, pages 487–504, 2011.

[MPN10] Avradip Mandal, Jacques Patarin, and Valérie Nachef. Indifferentiability beyond the birthday bound for the xor of two public random permutations. In *INDOCRYPT*, pages 69–81, 2010.

[MPS12] Avradip Mandal, Jacques Patarin, and Yannick Seurin. On the public indifferentiability and correlation intractability of the 6-round feistel construction. In *TCC*, pages 285–302, 2012. 9, 53

[NM08] Mridul Nandi and Avradip Mandal. Improved security analysis of PMAC. *J. Math. Cryptol.*, 2(2):149–162, 2008.