

論文 / 著書情報
Article / Book Information

題目(和文)	安全性証明におけるプログラム可能およびプログラム不可能なモデル
Title(English)	Programmable and non-programmable models in security proofs
著者(和文)	ラランジェラマリオ
Author(English)	Mario Larangeira
出典(和文)	学位:博士(理学), 学位授与機関:東京工業大学, 報告番号:甲第9051号, 授与年月日:2013年3月26日, 学位の種別:課程博士, 審査員:田中 圭介
Citation(English)	Degree:Doctor (Science), Conferring organization: Tokyo Institute of Technology, Report number:甲第9051号, Conferred date:2013/3/26, Degree Type:Course doctor, Examiner:
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

TOKYO INSTITUTE OF TECHNOLOGY

DOCTORAL THESIS

**Programmable and
Non-programmable Models in
Security Proofs**

Author:

Mario Rebello
LARANGEIRA Junior

Supervisor:

Dr. Keisuke TANAKA

*A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy*

in the

Tanaka Laboratory
Department of Mathematical and Computing Sciences

February 2013

Declaration of Authorship

I, Mario Rebello LARANGEIRA Junior, declare that this thesis titled, 'Programmable and Non-programmable Models in Security Proofs' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Mario Lorangeira Junior

Date: 02/10/2015

“For a successful technology, reality must take precedence over public relations, for nature cannot be fooled.”

Richard Feynman

TOKYO INSTITUTE OF TECHNOLOGY

Abstract

Graduate School of Information Science and Engineering
Department of Mathematical and Computing Sciences

Doctor of Philosophy

Programmable and Non-programmable Models in Security Proofs

by Mario Rebello LARANGEIRA Junior

We review the programmable and non-programmable variants of the Random Oracle Model (ROM), Generic Group Model (GGM) and Generic Ring Model (GRM) and their roles in security proofs.

More formally, this work recalls a new security proof for the RSA-FDH signature scheme in the non-programmable ROM under the one-more one-wayness assumption of the RSA trapdoor permutation.

Furthermore, we review four models: the *programmable and non-programmable* GGM and the analogous two for the GRM. We show that in the programmable generic models it is possible to turn around the negative result by Nielsen [1], regarding the non-interactive non-committing encryption (NINCE) in the presence of an adversary who adaptively corrupts the parties. This observation is evidence of the role of the programmability property, the intrinsic property of the programmable models, has in security proofs.

The use of such a property is illustrated by proving that the Goldwasser-Micali encryption scheme is NINCE scheme regarding corruption of the receiver in the programmable GRM. Whereas, for the programmable GGM, we show that the popular ElGamal encryption scheme is also NINCE despite the corruptions of the receiver and the sender. In both schemes the attack exposes the secret key.

Such observations contrasts with results in non-programmable models [1], where it is known that NINCE schemes are not possible, therefore are evidence of the crucial role that programmability has in security proofs.

Acknowledgements

Foremost, I would like to express my sincere gratitude to my advisor Professor Keisuke Tanaka for the continuous support, for his patience, motivation and enthusiasm. The happy, relaxed and, at the same time, serious and hard working environment he keeps in his laboratory still inspires me. That spirit, I believe, is crucial to all the Tanaka Lab members throughout their research activities.

In addition, I would like to thank my committee members, Professor Osamu Watanabe, Professor Sadayoshi Kojima, Professor Ken Wakita and Professor Ryo Kashima for their time and attention in listening the presentations and reading this PhD thesis. Their questions helped me to improve the contents of this work.

I must thank my fellow lab mates who since my arrival in Japan in 2006, when I was a research student, then during the master course, which I finished in 2008, and throughout the doctoral course, have been helping me, they are: Akihiro Yamada, Ryo Nishimaki, Christopher Portmann, Hitoshi Namiki, Takato Hirano, Toshiyuki Isshiki, Keita Xagawa, Toshihide Matsuda, Yuuki Tan, Akira Numayama, Hinako Kamimura, Naoyuki Yamashita, Daisuke Inoue, Kouichi Sakumoto, Ryotaro Hayashi, Chihiro Ohyama, Masatoshi Yashiro, Kousuke Nagumo, Hiroto-shi Takebe, Manh Nguyen, Yusuke Kinoshita, Takayuki Otsubo, Haruna Higo, Yoshihiro Koseki.

Special thanks to my friends that I met outside the laboratory. They too helped me to endure through the journey of the (sometimes not that easy) doctoral path. Their experience, being them Phd candidates or PhD degree holders themselves, their tips, and, of course, their support made the difference in the bad and good times. They are: Hercio Takashima, Mitsuhiro Fukuda, Aline Antunes, Sebastian Danilache, Lisa Ribeiro, Samir Ibradzic, Azra Sehic, Guilherme Ramos, Gleidson Pegoretti, Eyri Watari, Roberto Drebes, Fernanda Ushikubo, Leandro Teixeira.

I could not miss the opportunity to express my love and gratitude to my beloved Mario and Miriam, my parents, and my brother Diego. Moreover, I am also grateful for my aunties Maristela, Maria Regina and my grandmas Dilma and Julieta (which passed away last year), who even far away, every time we talk, state very clearly how proud they are of my achievements.

I also sincerely thank Eunice, my wife, for her understanding and seemengly endless love during all the time we have been together. A more mundane reason is all the times she helped me with documents in Japanese which, otherwise, would have made my life harder. For the last, to whom arrived for last, but whose toothless smile, that she shows to me in the end of a hard day, is the motivation of my life: my (almost) 6-month old daughter Sara.

Contents

Declaration of Authorship	ii
Abstract	iv
Acknowledgements	v
List of Figures	xi
List of Tables	xiii
Abbreviations	xv
Symbols	xvii
1 Introduction	1
1.1 Related Work	4
1.1.1 The ROM and its Programmability	4
1.1.2 The Generic Group and Ring Model	7
1.2 Main Topic	7
1.2.1 The Security of RSA-FDH in the Non-Programmable ROM	7
1.2.2 The Programmability in the Generic Group and Ring Models	10
1.3 Comparing ROM and Generic Models	12
1.4 General Definitions	13
1.5 Summary of the Chapters	14
1.5.1 Chapter 2	14
1.5.2 Chapter 3	14
1.5.3 Chapter 4	15
1.5.4 Chapter 5	15
2 The Non-programmable Random Oracle Model	17
2.1 Weakened Random Oracle Models	17
2.2 Preliminaries	19

2.2.1	One-way Trapdoors Permutations	19
2.2.2	The RSA Cryptosystem	20
2.2.3	Digital Signature Schemes	20
2.2.4	The Full Domain Hash Signature Scheme (FDH)	21
2.2.5	The EUF-CMA Security Game	21
2.2.6	The RSA-FDH	22
2.3	The Weakly Programmable Random Oracle Model	23
2.3.1	The WPROM and ROM	23
2.3.2	Black-Box Reduction in WPROM	25
2.4	Strengthening the One-way Trapdoor Permutation	26
2.4.1	The One-More Variant	26
2.4.2	The One-More Variant in WPROM	26
2.5	The RSA-FDH in the WPROM	27
2.6	A Tighter Security Reduction	32
2.7	Security Analysis	38
2.8	Summary	39
3	Programmability in the Generic Group Model	41
3.1	The Generic Group Model Evolution	41
3.2	Preliminaries	42
3.2.1	The Universal Composable Framework	42
3.2.2	The Non-Committing Encryption Protocol	43
3.2.3	The Non-Committing Encryption Functionality F_{nce}	44
3.2.4	The Schwartz Lemma	44
3.2.5	The Generic Group Algorithms	44
3.3	The Programmable and Non-Programmable GGM	45
3.3.1	The Programmable GGM	45
3.3.2	The Non-programmable GGM	46
3.4	The ElGamal Encryption Scheme in the Programmable GGM	46
3.4.1	The ElGamal Encryption Scheme	46
3.4.2	The Security of the ElGamal Encryption Scheme in the Programmable GGM	46
3.5	Summary	50
4	Programmability in the Generic Ring Model	53
4.1	The Generic Ring Model Evolution	53
4.2	Preliminaries	54
4.2.1	Straight Line Programs	54
4.2.2	Generic Ring Algorithms	55
4.3	The Programmable and Non-Programmable GRM	57
4.3.1	Programmable-GRM	57
4.3.2	Non-programmable-GRM	57
4.4	The Goldwasser-Micali Encryption Scheme in the Programmable GRM	57
4.4.1	The Goldwasser-Micali Encryption Scheme	58

4.4.2	The Security of the GM Encryption Scheme in the Programmable GRM	58
4.5	Summary	64
5	Final Remarks	65
	Bibliography	69

List of Figures

1.1	A typical proof with programmability in ROM.	2
1.2	In GGM when an algorithm $\mathcal{A}$ computes $a \circ b$, it submits queries with pointers to the values chosen by the generic group operation oracle G , not the values of a and b	3
1.3	The algorithm/adversary $\mathcal{A}$ submits queries $(op_{i_1}, op_{j_1}, \circ_1)$ to the generic operation oracle R , while it provides $\mathcal{A}$ with tokens/pointers $op_1, op_2, \dots, op_k$ and keeps track of the computation.	11
1.4	The simulator $\mathcal{S}$ programs the operation oracle R^{PRO}	12

List of Tables

1.1	Security reduction for the FDH in the programmable and non-programmable ROM, and the standard model.	8
1.2	Research on gaps and programmability for the ROM, the GGM and the GRM.	10
2.1	The weakened random oracles and the security of the schemes. . . .	18
2.2	Full Domain Hash signature scheme definition.	21
2.3	The Full Domain Hash signature scheme based on the RSA Cryptosystem.	22
2.4	Honest and adversarial interfaces of the WPRO.	24
3.1	Definition of the routine for the generic group operation oracle $\mathbf{G}$. . .	45
3.2	The ElGamal encryption scheme.	47
4.1	Definition of the routine for the generic ring operation oracle $\mathbf{R}$. . .	56
4.2	The Goldwasser-Micali encryption scheme.	58
4.3	Definition of the routines testpro and equalpro for the size of the heading $l_h(k)$ for protocol π and the simulated ciphertexts $R_{l_h(k)+r}$	60

Abbreviations

BB	B lack B ox
CCA	C hosen C iphertext A ttack
CMA	C hosen M essage A ttack
CT-ROM	C ollision T ractable R andom O racle M odel
EUUF	E xistential U nforgeability
FDH	F ull D omain H ash
FPT-ROM	F irst P re-image T ractable R andom O racle M odel
GM	G oldwasser M icali
GGM	G eneric G roup M odel
GRM	G eneric R ing M odel
GCD	G reat C ommon D ivisor
HYB	H Y B rid
NCE	N on- C ommitting E ncryption
NINCE	N on- I nteractive N on- C ommitting E ncryption
OAEP	O ptimal A symmetric E ncryption P adding
OM	O ne M ore
OW	O ne- W ay
PPT	P robabilistic P olynomial- T ime
QNR	Q uadratic N on- R esidue
QR	Q uadratic R esidue
REC	R E C eiver
RO	R andom O racle
ROM	R andom O racle M odel
SEN	S ENder

SLP	S traight L ine P rogram
SPT-ROM	S econd P reimage T ractable R andom O racle M odel
UC	U niversal C omposability
WP	W eakly P rogrammable
WPRO	W eakly P rogrammable R andom O racle
WPROM	W eakly P rogrammable R andom O racle M odel

Symbols

$\mathcal{A}$	Adversary or algorithm
$\mathcal{A}$	Interactive Turing Machine for the the adversary
1^k	Constant bit string $1 \dots 1$ of length k
$a \bmod N$	Remainder of a module N
F_{NCE}	Interactive Turing machine for the non-committing functionality
ℓ	Parameter of the one-more one-wayness assumption
σ	Digital signature
$\mathcal{G}$	Generator algorithm
G	Interactive Turing machine for the generic group operation oracle
$\mathcal{R}$	Domain set algorithm
R	Interactive Turing machine for the generic ring operation oracle
$\mathcal{O}$	Random oracle
$\mathcal{I}$	Inverter algorithm
$P_i \sqsubseteq P$	SLP P_i is a subset of the SLP P
$\ln$	Natural logarithm
$\mathcal{M}$	Message space
$\mathbb{N}$	Set of natural numbers: $\{1, 2, \dots\}$
$\mathbb{Z}$	Set of Integers
S	Interactive Turing machine for the simulator
Z	Interactive Turing machine for the environment
$\gcd(a, b)$	Greatest common divisor between a and b
$\text{negl}(x)$	Negligible function
q_h	Number of hash queries

q_s	Number of sign queries
$\rho(r)$	Honest interface of the weakly programmable random oracle
$J_N(x)$	Jacobi symbol x modulo N
p, q	Prime numbers
$\mathbb{Z}_N$	Residue class ring module N
$\mathbb{Z}_N^*$	Units in $\mathbb{Z}_N$
$\phi(N)$	Euler totient function
ε_{FACT}	Upper probability of any algorithm in solving the Factoring problem
ε_{OM}	Upper probability of any algorithm in solving the One-More Problem
π	Protocol

Dedicated to Eunice and Sara.

Chapter 1

Introduction

Provable security is a relatively new area of cryptography as noted by Koblitz and Menezes [2]. The first cryptosystem that could be proved secure appeared in 1979 by Rabin. This area embraces reasoning about the security of cryptographic schemes and it usually involves what is known as the reductionist argument.

In the heart of all reductionist security arguments lies a mathematical reduction which shows that the hardness of one problem P_1 implies the hardness of another problem P_2 . In other words, it implies that any algorithm which solves P_2 in reasonable amount of time could be used to solve P_1 . Thereby giving a guarantee that no one in the future will discover an algorithm capable of breaking a given cryptographic scheme, the problem P_2 , as long as the problem P_1 is assumed to be hard.

More technically, the proof is made of two parts besides the reduction algorithm: computational assumption and the assumed model regarding primitives and algebraic structures. In the cryptographic literature the following models appear very often: (1) the random oracle model (ROM), and (2) the generic group model (GGM) and, its generalization, the generic ring model (GRM).

These three models will be the subject of our study from now.

In the ROM methodology one considers a function $h : X \rightarrow Y$, for finite sets X and Y , as an ideal hash function. Moreover, it is common to choose arbitrary (but convenient for some security game) values for the outputs of the function h in security proofs of cryptographic schemes. This ability is denoted as the *programmability* of the random oracle model.

Very often proofs in the random oracle model use this ability. As an example, consider the notion of *existential unforgeability under the adaptive chosen message attack* (EUF-CMA) [3] by Goldwasser, Micali and Rivest. Briefly, in this notion the adversary is expected to compute a forged signature given a public key, and the access to both the random and the signing oracles. Once the adversary outputs a valid forgery of the form $\sigma = h(m)^d \bmod N$, i.e., the RSA-FDH signature scheme by Bellare and Rogaway [4], it is known that the value $h(m)$ must have been delivered to the adversary by the random oracle. In other words, the adversary cannot sample it by itself, since, in such a case, σ would be a *valid* forgery (the verification procedure returns 1) only with negligible probability due to the model. Similar ideas were used to construct security proofs in [5–7], i.e., the reduction algorithm embeds the challenge of the security proof into the outputs of the random oracle, therefore using the programmability.

The Figure 1.1 illustrates a typical proof in the ROM that uses programmability. Note that the reduction algorithm simulates a security game for the adversary, $\mathcal{A}$, while provides a simulation for the random oracle, in the figure $\mathcal{O}^{SIM}$.

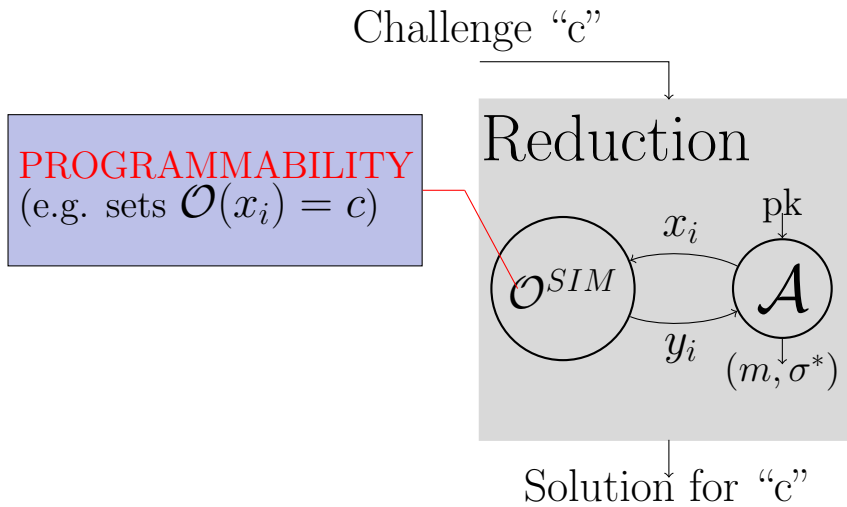


FIGURE 1.1: A typical proof with programmability in ROM.

The motivation for such a proof is that by embedding the challenge "c" into the hash queries the reduction can infer something from the output (m, σ^*) . In fact, very often, it can compute the solution for the challenge.

These observations received special attention after the work of Nielsen [1]. His result showed that there is a gap between the models with and without programmability.

On the other hand, the GGM, formally introduced by Shoup [8] and Nechaev [9], does not deal with hash function, but with the algebraic structure of the group used by the cryptographic scheme. It captures the situation that *no property* of the representation of the groups are available to be exploited. The only available features are the group operation $\circ$ and the equality test, i.e., $\{=, \circ\}$, between two members of the group. Both features are modeled as a generic operation oracle.

The concept of GGM is somewhat more difficult to grasp than that of the ROM. In order to better understand what it means, the reader should consider the weakest model we can conceive: the standard model. This model is what we consider as reality, that is, where there is no oracle to compute with. When an algorithm computes, for example, $a \circ b$ for group members a and b , it simply apply the group law $\circ$ to them.

In comparison, in GGM, the algorithm within the model submits operation query, which is a pair of pointers to a and b , to the generic operation oracle, say G , as illustrated by Figure 1.2. The answer to such a query is a pointer to the result of the operation.

It should be clear, that a major difference between GGM and the standard model is that, in GGM, the algorithm $\mathcal{A}$ does not receive the real values for a and b . Instead it receives pointers, or tokens, to them and the result of the operation. If it wants to continue the operation it should submit another query and so on.

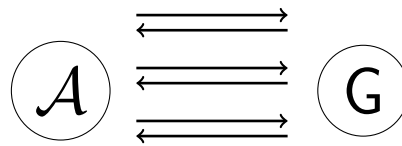


FIGURE 1.2: In GGM when an algorithm $\mathcal{A}$ computes $a \circ b$, it submits queries with pointers to the values chosen by the generic group operation oracle G , not the values of a and b .

The generalized notion of GRM models all the ring operations, i.e., $\{+, -, \times, /\}$, and tests for equality and existence of the inverse of the members of the ring.

Both GGM and GRM have been used along the years to provide evidence for intractability of computational problems and to investigate relations between problems, e.g., [8, 10, 11]. One of the most famous use of the GGM is in provable security, namely, in the two-step security proofs: (1) find a reduction, say B , from

a problem P to a successful adversary, and (2) show that the P is intractable in GGM. This strategy provides the guarantee that, at least in GGM, the challenge of breaking a scheme, represented by the successful adversary, is similarly intractable.

1.1 Related Work

There are several works regarding the ROM, GGM and GRM. They cover research about the models themselves, i.e., new formulations and weakened variants of them, as well as results regarding security of cryptosystems which rely on these models.

We start reviewing these works by the more popular among them: the Random Oracle Model.

1.1.1 The ROM and its Programmability

A main goal in cryptography is to implement adaptively secure protocols. As observed in [1], until now only statically secure implementation from the information theoretical model¹ are known. The idea is to implement them in the cryptographic model without erasure². This approach is based on Canetti's observation in [12, 13] that any CCA (*chosen ciphertext attack*) secure encryption scheme would work as a secure implementation as long as it has the *non-committing*³ property [13], and includes in the message two identification strings in order to protect against retransmission of the message by an attacker: one for the message (id_m) and the other for the sender (id_{SEN}).

A particular result about the ROM was discovered by Nielsen in [1]. He was the first to observe that a specific property of the ROM is required to prove security for an entire class of encryption algorithms, i.e., non-interactive and non-committing encryption (NINCE). The property name was coined the *programmability* of the ROM. Briefly, programmability is the power that B has to choose the output of the random oracle when interacting with the adversary (e.g., B chooses y arbitrarily

¹Where secure channels are assumed to exist.

²Where there are no secure channels and the parties are not trusted to erase their computation.

³Briefly, the encryption scheme that has the property of generating ciphertexts which can be opened to two different plaintexts.

and sets $h(x) = y$. That, ultimately, may help $\mathbf{B}$ in solving $\mathbf{P}$, giving the security proof. This result essentially means that there is no NINCE protocol in the standard model, since no hash function is known to exist with the programmability feature required by actual security proofs despite the efforts to construct this kind of primitives [14].

In a simulation-based security proof, as in [1], recall that the simulator $\mathcal{S}$ always programs the random oracle, both in the non-programmable and the programmable settings. It may seem contradictory, however the difference is that in the former, the *only good simulation* is when $\mathcal{S}$ repeats the value received from $\mathcal{O}$. Otherwise, the distinguisher easily distinguishes the executions by asking itself the same query directly to $\mathcal{O}$. Obviously, this argument does not work in the programmable setting (where the adversary cannot access $\mathcal{O}$ directly).

For almost seven years, since [1] the programmability of ROM remained as a understudied topic until recent work carried by Wee [15] and Fischlin et al. [16].

The latters were the first to introduce a framework to address the programmability on reductions, i.e., reduction-centric, instead of the simulation-based approach of the original model of Nielsen. They point out that this approach is indeed relevant, since even in the simulation-based proof one needs to construct a reduction from an instance problem to the distinguisher and this may require the programmability. Their framework is the best model in the literature to formally study proofs with and without programmability.

Their framework introduced the model named *weakly programmable random oracle model* (WPROM). Briefly, their new model prevents programmability by relying on the one-wayness of the one-way function ρ , by giving the adversary access to two interfaces. In one it receives the image $\rho(r)$, while in the other it receives the preimage r , both as the output of the random oracle. A reduction which tries to program the random oracle, can choose $\rho(r)$ but does not know r due to the one-wayness of ρ , therefore the model is non-programmable.

The authors of [16] remark that their model allows what they named *randomly programming*, that is, a reduction can still choose values for r . In comparison, their model addresses a restrictive sort of non-programmability, i.e., all reductions which program the oracle is expected to fail, due to the one-wayness of ρ , if they set directly the values of $\rho(r)$. One of the observations of their study is that this very specific restriction on the programmability in WPROM is enough to show that no

reduction exists to prove FDH secure. In particular no algorithm reduces efficiently from the one-wayness of the trapdoor permutation to an adversary breaking the FDH in their model.

In the *standard model*, i.e., the hash function is not the random oracle, Dodis, Oliveira and Pietrzak [17] previously studied the FDH scheme and found that, essentially, no property of the trapdoor permutation allows a security proof. Moreover it is not known whether any number-theoretic assumption would allow a proof of security.

The interest for the study of programmability, in particular for proofs that do not rely on programmability, is explained since no concrete hash function known so far has features which allow them to be programmed as it is often required in security proofs⁴. The consequence is that a security reduction may not be constructed in practice as observed in [16].

The results in the standard model [17] and in the non-programmable ROM [16] combined leave the security of FDH in state of uncertainty regarding the real construction of the reduction presented by the current security proofs [5, 18]. The practical implications of such real constructions are of prime importance since widely used cryptographic standards [19] for digital signatures rely on them.

Later, in this work, we will explore a different approach. Although the security of the RSA-FDH scheme perfectly holds in the ROM, the previous proofs do not provide a construction for the reduction because they are built in the programmable ROM. Since it is hopeless to look for a property of the trapdoor permutation in the standard model, an alternative is to look for it in the non-programmable ROM. Chapter 2 reviews the, so far, only non-programmable security proof for the RSA-FDH, which means that at least in the Fischlin’s model the non-programmable security is possible. We see this novel approach as an effort to put the security of the RSA-FDH on more firm grounds.

⁴Programmable hash functions were introduced by Hofheinz and Kiltz [14], but unfortunately their hash functions still lack the level of programmability required in the regular proofs of security.

1.1.2 The Generic Group and Ring Model

Since their introduction, in the Nineties, they have been used to argue about the assumed hardness of computational problems, i.e., computing discrete logarithms, the computational Diffie-Hellman problem, the decisional Diffie-Hellman problem and etc. Maurer introduced a different formulation [11], which Jager and Schwenk showed to be equivalent to Shoup's formulations [20] in a work which studies the models themselves. In 2008, the GGM was generalized to the GRM by Aggarwal and Maurer [10] using ideas already proposed by Boneh and Venkatesan [21]. Based on [10], Jager and Schwenk [22] improved once more the formulation of the GRM and showed that there are problems that can be solved efficiently in practice, however in the GRM they are equivalent to factoring the modulus $N = p \cdot q$.

More recently, the study of problems in the generic models has been improved by the introduction of a new model named the semi-generic group model by Jager and Rupp [23]. This new model aims to cover the recently increased number of problems specially after the advent of pairing cryptography.

1.2 Main Topic

In general, this work aims to investigate the models considered in provable security. Our findings cover schemes in the already mentioned ROM, GGM and GRM.

1.2.1 The Security of RSA-FDH in the Non-Programmable ROM

The main finding for the RSA-FDH in ROM is that a non-programmable security reduction in the Fischlin's WPRM is possible. It may sound to the reader that it is a contradiction to the previous work, therefore we remark that for the security proof we also propose a strengthening of the one-way assumption for the trapdoor permutation. We adapted the one-more RSA assumption introduced by Bellare et al. in [24] to the WPRM.

Although this notion is not as weak as the standard notion, which often is used for signature schemes, it is a natural and reasonable generalization for the classical

Trapdoor permutations			
$\forall$ property (essentially)	One-way		One-more One-way
Standard model	non-prog. ROM	prog. ROM	non-prog. ROM
$\nexists$ security reduction [17]	$\nexists$ security reduction (Fischlin's Model) [16]	$\exists$ security reduction (theoretical reduction) [5, 18]	$\exists$ security reduction (Fischlin's Model) [Section 2.2] [25]

TABLE 1.1: Security reduction for the FDH in the programmable and non-programmable ROM, and the standard model.

one-wayness. Furthermore, we show that combined with the randomly programming of Fischlin's WPRM, it brings the advantage to permit the construction of a non-programmable security proof based on the random oracle model.

In a theoretical point of view, the result by Larangeira and Tanaka [25] is relevant, since it shows a middle ground between the general result of [17] and the one in [16]. Moreover, the result in [25] still relies on crucial properties of the random oracle described earlier, i.e., when the adversary outputs a valid forgery, it can be assumed that the hash of the message was delivered by the random oracle. Thus, a conclusion is that, in random oracle based proofs, this property may play a more crucial role comparing to the programmability of the ROM for proving the security of RSA-FDH scheme.

Later, we review two theorems about the security of RSA-FDH regarding the number of hash and sign queries, respectively, q_h and q_s , and ℓ , where ℓ is the parameter for the one-more one-wayness assumption. In particular we present the following

- Naive approach: q_h and q_s , are less than ℓ (Theorem 2.18, Lemmas 2.19 and 2.20).
- Tighter version: $q_h + q_s \geq \ell$, (Theorem 2.22 and Lemma 2.21) and the security analysis regarding q_s , ℓ and the security parameter k in Section 2.7. That is, whenever $q_s < \ell - k \cdot \ln(2)$ the RSA-FDH is EUF-CMA secure.

The proofs are based in two disjoint cases: when a forgery would be generated by (1) the (unlikely) collision found by the adversary using the (weakly programmable) random oracle, and (2) a collision have not happened and the adversary inverted, by itself, the trapdoor permutation. We focus our analysis in (2) (Lemma 2.20 and Lemma 2.21).

Recall that in the standard one-wayness of the trapdoor permutation f_{pk} , an algorithm, say, $\mathcal{I}$, receives a random value from the domain and the public key pk , and is required to output $z = f_{\text{sk}}^{-1}(y)$. The approach using this property is not suitable for the non-programmable setting since in the classical EUF-CMA the adversary receives several points sampled by the random oracle.

Our approach is to consider a definition more similar to the model of attack (i.e., EUF-CMA) and more general, since in [16] only the particular case of *none* signature queries was considered.

That is the adversary and the reduction receive arbitrary points in the domain of the trapdoor but choose one to compute the inverted value for the trapdoor on that point (i.e., construct the forgery as in (2) in our analysis) after received at most $q - 1$ inverted values (i.e., signature queries). This translates into a property similar to the one-more RSA assumption [24], that is the inverter algorithm $\mathcal{I}$ to query the inverter oracle f_{sk}^{-1} at most $\ell - 1$ times.

The challenge starts by $\mathcal{I}$ receiving the pk and ℓ values r_i . It is expected to output ℓ inverted values $z_1, \dots, z_\ell$, such that $f_{\text{pk}}(z_i) = y_i$ with at most $\ell - 1$ queries to the inverter oracle f_{sk}^{-1} , for uniformly random y_i chosen by the random oracle $\mathcal{O}$. Adapting it to WPROM, this means that y_i are $\rho(r_i)$, where ρ is defined by the model.

Note that $\mathcal{I}$ is not expected to invert a single value as in the classical one-wayness notion. Instead, it chooses among the received *arbitrary* values y_i sampled randomly from the domain. Needless to say that $\mathcal{I}$ cannot choose the points y_i , which are chosen by the random oracle $\mathcal{O}$.

1.2.2 The Programmability in the Generic Group and Ring Models

While several proofs in ROM rely on programmability, until the work of Larangeira and Tanaka [31], such property had not been formalized for generic group and ring models. Later chapters investigate the role of such a property in security proofs in the generic group and ring models.

In order to introduce the formal study of programmability in these models, it is necessary to review the weakened variants of the generic group and ring models, the programmable and non-programmable GGM (and GRM). Therefore the research about this models embraces four models: programmable GGM and GGM, and non-programmable GGM and GRM.

Before [31], the study of programmability in the generic models does not seem to have been proposed, although the idea of “programming the oracle” is not new, and even similarities, as *gaps* between the models and real situations, have been observed in the ROM and the GGM, e.g., Canetti, Goldreich and Halevi [26] and Dent [27]. Hence from a theoretical viewpoint, this work partially fills a gap regarding generic models in the current status of the research as shown in Table 1.2.

	ROM	GGM	GRM
Gap	[26, 28]	[27, 29, 30]	[22]
Programmability	[1, 15, 16]	[Section 3.2] [31]	[Section 4.2.1] [31]

TABLE 1.2: Research on gaps and programmability for the ROM, the GGM and the GRM.

The intuition of our models comes from the two-step approach for security proofs. Recall that the lower bound in the complexity in the GRM and the GGM only applies for algorithms that can be cast in these models. Therefore, although the reduction B can be general, i.e., for all classes of algorithms, the security may not be carried from the generic models.

Implicitly, these proofs assume that every party has access to the generic group (or ring) operation oracle. Our models explicitly capture this feature by using the G-hybrid model from the universal composable (UC) framework [12] for the

GGM (and R-hybrid model for the GRM). This makes our programmable models equivalent to the regular GGM [8, 9] and GRM [10, 22].

The difference between the two models we propose is analogous to the Nielsen’s programmable and non-programmable ROM. For example, in the case of the GGM, we make G widely available in the non-programmable GGM while *not* in the programmable counterpart.

Recall that the generic operation oracle collects the operations as steps, formally as a tuple $(i, j, \circ)$, where $\circ$ is the group operation (or one of the ring operations) and i and j are pointers for previously computed steps as illustrated in Figure 1.3.

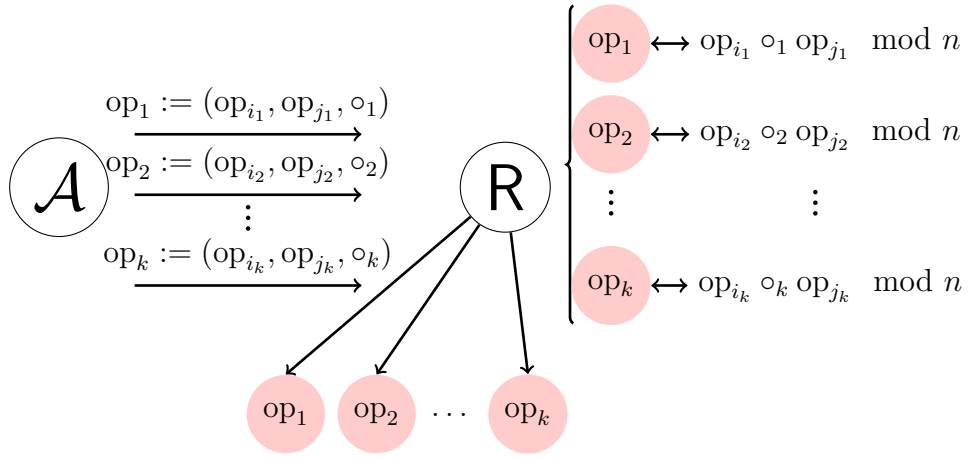


FIGURE 1.3: The algorithm/adversary $\mathcal{A}$ submits queries $(op_{i_1}, op_{j_1}, \circ_1)$ to the generic operation oracle R , while it provides $\mathcal{A}$ with tokens/pointers $op_1, op_2, \dots, op_k$ and keeps track of the computation.

We propose the use of these steps to “program” the generic operation oracle similarly to the random oracle model. That is, when the simulator, say $\mathcal{S}$, in a simulation-based proof of security, needs to simulate the ciphertext between the parties without knowing the message m , it delivers steps with *no computation associated with it* as the ciphertext.

This approach allows the simulator, after learning m , to associate a suitable computation consistent with m , therefore creating a non-committing ciphertext while preserving a perfect simulation.

Figure 1.4 illustrates the interaction between the algorithm/adversary $\mathcal{A}$ and the simulator $\mathcal{S}$ through the operation oracle R^{pro} provided by the simulator. In blue it is the computation that needed to be adjusted after knowing the message m .

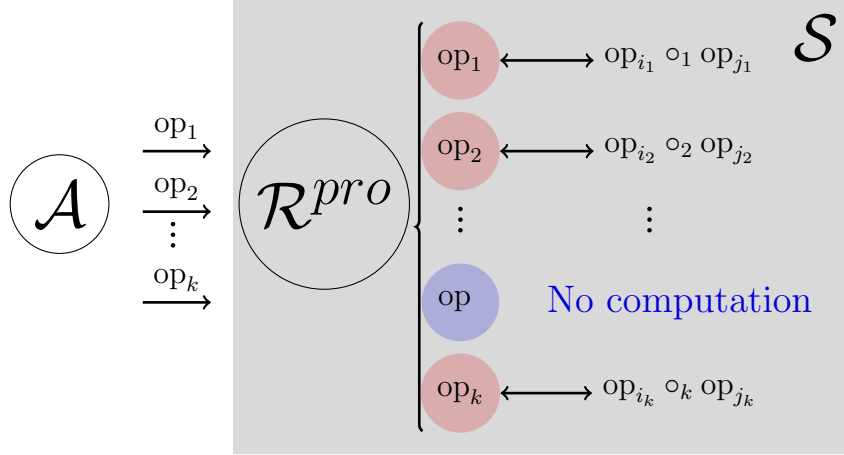


FIGURE 1.4: The simulator $\mathcal{S}$ programs the operation oracle R^{PRO} .

1.3 Comparing ROM and Generic Models

We observe that while the ROM is a more accepted model to argue about the security of cryptographic schemes, some skepticism is levered against generic models [27, 29]. However, in the case of the GGM, some researchers claim that the issues raised along the years are, in fact, due to issues in the proofs instead of the models and not from weaknesses of the GGM, e.g., [30, Section 5], thus the model is as worth study as the ROM.

We recall that in the ROM, it is hopeless to use the model to try to justify the security of a system when a broken hash function is considered as a substitute for the random oracle (think, for example, the already broken MD5 hash function for signature schemes).

Analogously, for the case of the GGM, it is hopeless to use the model to prove the security of schemes when the concrete instantiation of the group is a “bad” choice. Consider, for example, a “bad” choice means the groups that are known to be susceptible to the use of the index calculus algorithm⁵, or any other algorithm that gives a much faster and potential alternative for adversaries which does not fit in the GGM.

Another common concern about proofs in generic models is that it applies to a restricted class of *generic* adversaries, i.e., adversaries that do not receive a description of the concrete group being considered. We emphasize that the same is true for proofs in the ROM. Recall, that the adversary in ROM does not receive

⁵It is known that the index calculus algorithm cannot be cast in the GGM.

the description of a hash function, whereas in an implemented system, i.e., when a good hash function is used instead of the random oracle, the security analysis must assume that the description of the hash function is widely available.

To strengthen our analogy, in this work, we see the ROM as a “generic hash function model”. That is, a model where the adversary treats the hash function as a black box, much in the same way as for the generic group operation oracle used in the GGM (and the analogous generic ring operation oracle in the case of GRM).

Thus, if the proofs that rely on the programmability of the ROM are accepted as reliable security guarantees for a family of good hash functions, then we believe the same should be true for the case of “good” cyclic groups. Where “good” means groups whose properties beyond the group laws are not known to exist (certain elliptic curves groups, for example).

In comparison to the GGM, not too many schemes are known to have security proofs exclusively in the GRM. Furthermore, results like those in [31] shows that a similar technique applies to the GRM regarding the use of programmability, although the model has been showed not to be soundness [22].

The observations in the latter chapters regarding programmability in the GGM and the GRM, likewise those in the ROM, show that programmable and non-programmable variants of the models are fundamentally different. Moreover, although based on stronger models (programmable GGM and GRM), they are as valid as proofs (with programmability) in ROM.

1.4 General Definitions

In this work, PPT stands for *probabilistic polynomial-time*. When a pair (y, x) is said to be defined, it means that the value x has already been queried to the random oracle and y was returned. It is said that a function f is negligible with respect to k , i.e., $\text{negl}(k)$, whenever for any $c, d \in \mathbb{N}$ there exists $k_0 \in \mathbb{N}$ such that for all $k > k_0$ and for all $x \in \{0, 1\}^{k^d}$, it holds that $f(x, k) < k^{-c}$.

Furthermore, gcd stands for the greatest common divisor and when x is uniform randomly chosen from some set X , it is written $x \xleftarrow{U} X$. Finally, assume that

$\mathcal{M} = \{0, 1\}^l$ is the message space for the scheme and k is the security parameter, and therefore $l_m(k)$, $l_r(k)$ and $q(k)$, respectively the number of bits of the message, the randomness and the number of the queries to the generic operation oracle, are polynomials of k .

1.5 Summary of the Chapters

In the next three chapters, Chapters 2, 3 and 4, we review the main models and results regarding, respectively, the Non-programmable Random Oracle Model, the Generic Group Model and the Generic Ring Model. In these chapters we discuss, and introduce, in the case of Chapters 3 and 4, the *programmability* of each model. In the final chapter, Chapter 5, we present our conclusion regarding this work.

From now, we present a more detailed summary of each chapter individually.

1.5.1 Chapter 2

We review the best framework to study programmability in ROM, the Fischlin's model [29], as well as other weakened variants of ROM. We also review the novel security proof for the RSA-FDH in the non-programmable ROM due to Larangeira and Tanaka [25].

In the next two chapters we recall novel models: the programmable and the non-programmable GGM in Chapter 3, and the analogous GRM versions, the programmable and the non-programmable GRM in Chapter 4.

1.5.2 Chapter 3

Section 3.4.2 illustrates our proof technique using the programmability by analyzing the ElGamal encryption scheme in the programmable GGM. Briefly, the proof gives a construction for a simulator and uses the Schwartz Lemma [8, 32] to find a lower bound in the distinguishing game. Thereby showing that ElGamal encryption scheme is NINCE under corruption of the sender and the receiver.

This chapter differs from the previous one because from this point, the proofs are simulation based, in comparison with the reduction-centric style used earlier. Therefore, we also review the Universally Composable Framework [12] which is used again in the next chapter.

1.5.3 Chapter 4

We give a brief review of the history of GRM and its evolution. Moreover, in Section 4.4.2, we rely on the formulation and machinery for the GRM by Jager and Schwenk [22] to show that the Goldwasser-Micali (GM) is NINCE scheme regarding only receiver corruptions in the programmable GRM. This result does not rule out the possibility that a NINCE scheme considering also sender corruptions exists, however it is sufficient to circumvent the result in [1] in the computational setting.

1.5.4 Chapter 5

Here we state a few final remarks regarding the results presented in the last three chapters. Briefly, our observations confirm the intuition that models with programmability are, indeed, stronger, however, in the case no other weaker model is available, this property can be used to provide guarantees that are valid in the realm of provable security, much in the same way the programmability in ROM has been used to argue about the security of cryptosystems.

Chapter 2

The Non-programmable Random Oracle Model

In this chapter, we give a brief overview of the weakened variants of the ROM, including the main subject of the chapter: the non-programmable ROM. Next, we show how to adapt the ℓ -One-more One-way trapdoor permutation property, the generalization of the regular one-wayness property, to the Fischlin et al. model [16]. In addition, we show that the RSA-FDH is EUF-CMA secure in the non-programmable ROM assuming the already mentioned one-more one-way property of the trapdoor permutation adapted to WPRM. The proof is composed by two theorems: the naive approach, when q_h and q_s , are less than ℓ , Theorem 2.18, Lemmas 2.19 and 2.20, and the tighter version, when $q_h + q_s \geq \ell$, Theorem 2.22 and Lemma 2.21. We also give the security analysis regarding q_s , ℓ and the security parameter k in Section 2.7 which completes the security proof.

2.1 Weakened Random Oracle Models

Several evidences point that the random oracle model (ROM) is a strong model [26] and there still much debate on the security guarantee provided by the ROM [33]. Therefore there is a line of research which the goal is to devise weaker models and study the security of cryptographic schemes under these new models.

- Tractable Weakened ROM
- Leaky ROM
- Auxiliary-Input ROM
- Explicitly and Weakly Programmable ROM

Tractable Weakened ROM. Numayama et al. [7] devised weakened random oracle models, respectively, CT-ROM (Collision Tractable), SPT-ROM (Second-Preimage Tractable) and FPT-ROM (First-Preimage Tractable). The idea of this work is to provide the adversary with specific oracles which break properties (i.e., *collision*, *second preimage* and *first preimage* resistances). This is equivalent to say that the random oracle does not have these properties in each one of these models.

In these new models four variants of the RSA signature scheme were studied: RSA-FDH [4], RSA-PFDH [6], RSA-PFDH⁺ [7] and RSA-PFDH[⊕] [7]. The results are summarized in Table 2.1. There is a follow up of this work by Kawachi [34], which

scheme/model	ROM	CT-ROM	SPT-ROM	FPT-ROM
RSA-FDH	secure	insecure		
RSA-PFDH	secure		insecure	
RSA-PFDH ⁺	secure			insecure
RSA-PFDH [⊕]	secure			

TABLE 2.1: The weakened random oracles and the security of the schemes.

studies encryption schemes regarding these new models.

Leaky ROM. Yoneyama et al. [35] devised a model to address the issue of the sloppy use of hash functions. The Leaky Random Oracle Model they introduced allows the adversary to extract arbitrary lists of inputs and outputs of the random oracle. Still, they studied the security of several cryptographic schemes, such as FDH, OAEP, Cramer-Shoup, Kurosawa-Desmedt regarding this new model. Their main conclusion is that this new ability of the adversary affects the security of the studied schemes. In particular, they showed that FDH is secure under this model.

Auxiliary-Input ROM. Another different aspect of ROM was analyzed by Unruh [36]. He devised a weaker model to study the role of the *auxiliary-input* in the ROM. His conclusion, based on this auxiliary-input ROM, is that an a particular technique lazy-sampling, which is widely employed in the regular ROM, can still be used in the presence of the auxiliary-input.

Explicitly and Weakly Programmable ROM. The already mentioned work of Nielsen [1, 37] and its suggested non-programmable ROM is another example of study about weakened ROM. That was the first work to study the *programmability* property of ROM. Different models, regarding different levels of programmability, were suggested by Wee [15] and Fischlin et al. [16].

Up to this point, the work of Fischlin et al. and its Weakly Programmable ROM (WPROM) is the more complete framework to study the programmability of ROM. Therefore we will consider this model to study the RSA-FDH signature scheme in a weaker variant of ROM: the non-programmable ROM.

2.2 Preliminaries

At this point, it is convenient to recall a few basic definitions.

2.2.1 One-way Trapdoors Permutations

Definition 2.1 (Collection of one-way trapdoor permutations). The tuple $(\mathcal{K}, \mathcal{F}, \mathcal{G}, \mathcal{R})$ is a collection of one-way trapdoor permutations with security parameter k , if $\mathcal{K}$ is an infinite index set, $\mathcal{F} = \{f(\mathbf{pk}, \cdot) = f_{\mathbf{pk}}(\cdot) : D_{\mathbf{pk}} \rightarrow D_{\mathbf{pk}}\}_{\mathbf{pk} \in \mathcal{K}}$ is a set of permutations for a domain $D_{\mathbf{pk}}$, the key/trapdoor-generator $\mathcal{G}$ and the domain-generator $\mathcal{R}$ are PPT algorithms, and the following holds:

Easy to compute (Generator algorithm): $\mathcal{G}(1^k) \rightarrow (\mathbf{pk}, \mathbf{sk})$ generates pairs of keys and trapdoors, where $\mathbf{pk} \in \mathcal{K} \cap \{0, 1\}^{p(k)}$ for some fixed polynomial $p(k)$. Nonetheless $f_{\mathbf{pk}}(\cdot)$ is PPT for $r \in D_{\mathbf{pk}}$ and computes $f_{\mathbf{pk}}(r)$.

Easy to sample (Domain set algorithm): $\mathcal{R}(\mathbf{pk}) \rightarrow r$, and r is uniformly random in $D_{\mathbf{pk}}$.

Hard to invert: For any PPT algorithm $\mathcal{A}$ the probability over the randomnesses of the algorithms $\mathcal{G}$ and $\mathcal{R}$ that $\mathcal{A}(\mathbf{pk}, f_{\mathbf{pk}}(r)) = r$ is negligible in k .

Easy to invert with the trapdoor: There is a PPT algorithm which computes

$$f^{-1}(\mathbf{sk}, f_{\mathbf{pk}}(r)) := f_{\mathbf{sk}}^{-1}(f_{\mathbf{pk}}(r)) = r,$$

for all $\mathcal{G} \rightarrow (\mathbf{pk}, \mathbf{sk})$ and $r \in D_{\mathbf{pk}}$.

2.2.2 The RSA Cryptosystem

One of the most popular candidates for collection of trapdoor (one-way) permutations is presented in the next definition.

Definition 2.2 (RSA collection [38]). Let the tuple $(\text{RSA}, D_{\text{RSA}}, F_{\text{RSA}})$ be called RSA collection with security parameter $k \in \mathbb{N}$, if the key/trapdoor-generation algorithm RSA and the domain-generator D_{RSA} are PPT algorithms in k , and the following holds.

- RSA , on input 1^k , outputs except with negligible probability a modulo N and two integers e and d , such that $e \cdot d = 1 \pmod{(p-1) \cdot (q-1)}$, for two k -bit long prime numbers p and q whose product is N .
- $D_{\text{RSA}}(N, e)$ samples an element from $\{1, \dots, N\}$.
- $F_{\text{RSA}} = \{f((N, e), \cdot) : D_{\text{RSA}}(N, e) \rightarrow D_{\text{RSA}}(N, e)\}$, with $f((N, e), x) = x^e \pmod N$.

2.2.3 Digital Signature Schemes

A digital signature is a string dependent of a message m and a secret value $\mathbf{sk}$ only known to the signer. Anyone can verify the signature given the public key $\mathbf{pk}$, the message m and the signature.

Definition 2.3 (Signature scheme). The tuple $(\text{gen}, \text{sig}, \text{ver})$ is a digital signature scheme with respect to a message space $\mathcal{M}$ if the three above mentioned algorithms are as follows.

- The key generation algorithm $gen(1^k)$ outputs the secret/public key pair $(\mathbf{pk}, \mathbf{sk})$ given the security parameter k . That is $gen(1^k) \rightarrow (\mathbf{sk}, \mathbf{pk})$.
- The sign algorithm $sig_{\mathbf{sk}} := sig(\mathbf{sk}, \cdot)$ output the signature σ when given the message $m \in \mathcal{M}$. That is $sig_{\mathbf{sk}}(m) \rightarrow \sigma$.
- The verification algorithm $ver_{\mathbf{pk}} := ver(\mathbf{pk}, \cdot, \cdot)$ outputs a bit b such that $b = 1$ if the signature σ is accepted with respect to m and $\mathbf{pk}$, otherwise b equals to 0. That is $ver_{\mathbf{pk}}(m, \sigma) \rightarrow b$.

2.2.4 The Full Domain Hash Signature Scheme (FDH)

The FDH is the signature scheme $(gen, sig_{\mathbf{sk}}, ver_{\mathbf{pk}})$ where the algorithms $sig_{\mathbf{sk}}$ and $ver_{\mathbf{pk}}$ have oracle access to a hash function $h : \{0, 1\}^l \rightarrow \{0, 1\}^k$ and $gen(1^k) \rightarrow (\mathbf{pk}, \mathbf{sk})$.

$sig_{\mathbf{sk}}(m)$	$ver_{\mathbf{pk}}(m, \sigma)$
$h(m) \rightarrow y$	$f_{\mathbf{pk}}(\sigma) \rightarrow y'$
$f_{\mathbf{sk}}^{-1}(y) \rightarrow \sigma$	if $h(m) = y'$
return σ	return 1
	else
	return 0

TABLE 2.2: Full Domain Hash signature scheme definition.

Next, we review the classical security game for digital signature schemes.

2.2.5 The EUF-CMA Security Game

This is the standard security game for digital signature schemes.

Definition 2.4 (EUF-CMA security game [3]). Given a signature scheme S , the existential forgery under adaptive chosen message attack game is defined by the following.

Game_{EUF-CMA}^h(k) :

1. The generation algorithm $gen(1^k)$ is run to obtain the keys $(\mathbf{sk}, \mathbf{pk})$.

2. Adversary $\mathcal{A}$ is given pk .
3. The adversary submits hash queries by sending x_i , and receives the values $h(x_i)$.
4. The adversary submits sign queries by sending m_j , and receives the values q_j .
5. Eventually, $\mathcal{A}$ outputs a forgery signature σ^* for m^* .
6. If $\forall j, m^* \neq m_j$ and $\text{ver}_{\text{pk}}(m^*, \sigma^*) = 1$, $\mathcal{A}$ wins the game and therefore output 1. Otherwise $\mathcal{A}$ loose, consequently output 0.

Definition 2.5. For an adversary $\mathcal{A}$ and a signature scheme $\mathcal{S}$, we define the advantage of $\mathcal{A}$ in the above game, written $\text{Adv}_{\mathcal{S}h}^{\text{EUF-CMA}}(\mathcal{A})$, as follows

$$\text{Adv}_{\mathcal{S}h}^{\text{EUF-CMA}}(\mathcal{A}) = \Pr[\text{Game}_{\text{EUF-CMA}}^h(k) \rightarrow 1]$$

where the probability is taken over the randomness of the game.

2.2.6 The RSA-FDH

Now we are ready to properly review the RSA-FDH signature scheme¹ which is given by the table below. The security of the RSA-FDH is based on the assump-

$\text{gen}(1^k)$	$\text{sig}_{\text{sk}}(m)$	$\text{ver}_{\text{pk}}(m, \sigma)$
$\text{RSA}(1^k) \rightarrow (N, e, d)$ $(N, e) \rightarrow \text{pk}$ $(N, d) \rightarrow \text{sk}$ return (pk, sk)	$h(m) \rightarrow y$ $y^d \rightarrow \sigma$ return σ	$\sigma^e \rightarrow y'$ if $h(m) = y'$ return 1 else return 0

TABLE 2.3: The Full Domain Hash signature scheme based on the RSA Cryptosystem.

tion that the RSA Cryptosystem, as in Definition 2.2, is a collection of one-way trapdoor permutations, as in Definition 2.1, in the (programmable) ROM.

Next, we review the WPRM, in particular the ρ -WPRM, therefore it is possible to choose a proper function ρ which gives the non-programmable ROM.

¹Consider $h : \{0, 1\}^l \rightarrow \{0, 1\}^k$ and exponentiations are done module N .

2.3 The Weakly Programmable Random Oracle Model

Here we review the WPROM. In particular we review the definitions for ρ -WPRO and the fully-black-box (BB) WPROM reduction from [16].

2.3.1 The WPROM and ROM

For comparison, we start the section by recalling the standard definition for the random oracle $\mathcal{O}$.

Definition 2.6 (Random oracle $\mathcal{O}$). The parties (and the adversary) are allowed to evaluate values $x \in \{0, 1\}^l$ and receive the value $z \in \{0, 1\}^k$ chosen uniformly at random independent of previous evaluations, except for the evaluation on the same value x , in which $\mathcal{O}$ returns z again.

In order to move to the WPROM, first we need to review the definition for GOOD function ρ .

Definition 2.7 (GOOD function ρ [16]). Given two sets **COINS** and **RNG**, a function $\rho : \text{COINS} \rightarrow \text{RNG}$ is named GOOD for **RNG**, if and only if

1. **COINS** is a finite set
2. $|\text{RNG}|$ divides $|\text{COINS}|$
3. ρ is regular, that is $\forall y \in \text{RNG}$

$$|\{r \in \text{COINS} : \rho(r) = y\}| = \frac{|\text{COINS}|}{|\text{RNG}|}.$$

From now we consider $\text{RNG} = \{0, 1\}^k$ and $\text{COINS} = \{0, 1\}^c$ for a fixed integer c and the security parameter k as given by the RSA-FDH scheme from Section 2.2.6.

Now we need to review the weakly programmable random oracle (WPRO).

$\mathcal{O}_{hon}^\rho(x) :$ if $\exists$ a pair $(z, x) \in \mathsf{T}$ return z else sample $r \leftarrow \mathsf{COINS}$ set $z \leftarrow \rho(r)$ add (z, x) in T add (r, x) in R return z	$\mathcal{O}_{adv}^\rho(x) :$ if $\exists$ a pair $(z, x) \in \mathsf{T}$ return z and r (such that $(r, x) \in \mathsf{R}$) else sample $r \leftarrow \mathsf{COINS}$ set $z \leftarrow \rho(r)$ add (z, x) in T add (r, x) in R return z and r
--	--

TABLE 2.4: Honest and adversarial interfaces of the WPRO.

Definition 2.8 (ρ -WPRO [16]). Given a GOOD function ρ , and two pairwise tables T and R , the weakly programmable random oracle $\mathcal{O}^\rho$, or ρ -WPRO, is composed by the honest and the adversary interfaces, that is $\mathcal{O}^\rho = (\mathcal{O}_{hon}^\rho, \mathcal{O}_{adv}^\rho)$, where the two interfaces are defined in Table 2.4

Remark 2.9. As an example, if the adversary $\mathcal{A}$ submit a hash query on x it would receive two values, namely r and $\rho(r)$, through the adversarial interface $\mathcal{O}_{adv}^\rho$.

Remark 2.10. As discussed in [16], depending on the properties of ρ , the WPROM may vary from non-programmable to fully-programmable ROM. In particular,

- If ρ is the identity function, then WPROM is equivalent to the programmable ROM.
- If ρ is one-way function, then WPROM is non-programmable ROM.

It is easy to see that when ρ is one-way function it becomes difficult for any reduction to program the random oracle. In particular, any PPT reduction, given a value z' for the output of the random oracle, will find difficult to compute $\rho^{-1}(z')$ in order to find a suitable r' . Since we are interested in the non-programmable ROM, it is convenient to give an extra definition.

Definition 2.11 (GOOD_{OW} function ρ). A function $\rho : \mathsf{COINS} \rightarrow \mathsf{RNG}$ is said to be GOOD_{OW} if it is GOOD for RNG and it is also one-way function.

Remark 2.12. From now, assume for the rest of the work that ρ is GOOD_{OW} function.

It is also known from [16, Theorem 4.6], that ROM and WPRO are equivalent models. In addition that being a weakly programmable random oracle is in fact a *weaker requirement* than being a full-fledged random oracle, in the sense that it is possible to construct a random oracle out of a WPRO.

From now, we can properly give the formal definition for the fully-black box (BB) security reduction in WPRO, i.e., WPRO reduction for short.

2.3.2 Black-Box Reduction in WPRO

Given a cryptographic scheme S with access to $\mathcal{O}_{hon}^\rho$, and which uses a primitive f , written $S^{\mathcal{O}_{hon}^\rho}[f]$, a reduction $\mathcal{B}$ that breaks the property π' of the primitive f , using an adversary $\mathcal{A}$, which, respectively, attacks the property π of the scheme S is formalized in the following definition.

Definition 2.13 (WPRO reduction [16]). For $q_{\mathcal{A}}$ instances of the adversary $\mathcal{A}$, q_ρ queries to ρ , q_f queries to f and running time t , the $(\pi' \leftarrow \pi, \delta, t, q_\rho, q_f, q_{\mathcal{A}})$ -fully-BB WPRO security reduction for S is an oracle machine $B^{(\cdot, \cdot)}$ with the property that for all π -adversaries $\mathcal{A}^{(\cdot)}$, all GOOD functions ρ for RNG, and all π' -candidates f , if

$$Adv_{S^{\mathcal{O}_{hon}^\rho}[f]}^\pi(\mathcal{A}^{\mathcal{O}_{adv}^\rho}) > \varepsilon$$

for ρ -WPRO $\mathcal{O}^\rho = (\mathcal{O}_{hon}^\rho, \mathcal{O}_{adv}^\rho)$

$$Adv_f^{\pi'}(B^{(\rho, f, \mathcal{A}^{(\cdot)})}) > \delta(\varepsilon, q_{\mathcal{A}}, q_\rho, q_f, t).$$

The intuition is that given an adversary $\mathcal{A}$ with minimal success probability ε in breaking π , then the reduction B with oracle access to ρ , f and $\mathcal{A}$ in WPRO has minimal success probability δ in breaking π' .

As pointed in Remark 2.10, if ρ is GOOD_{OW} the reduction B will find very difficult to simulate the *adversarial interface* $\mathcal{O}_{adv}^\rho$, i.e., the pair (z', r') , for an arbitrary z' and $r' = \rho^{-1}(z')$. Therefore, very likely, the success probability of B is negligible in the non-programmable setting.

Next, we describe our approach to give a stronger property for the collection of one-way trapdoor permutation of Definition 2.1.

2.4 Strengthening the One-way Trapdoor Permutation

In this section we give the strengthening of the regular one-way trapdoor permutation assumption adapted to the WPRM.

2.4.1 The One-More Variant

Here we give the strengthening of Definition 2.1.

Our notion for the one-wayness resembles the one considered in [24] and [39] based on the one-more RSA problem, i.e. ℓ -OM-RSA, which suggests that the RSA collection of trapdoor (one-way) permutations satisfies this extra requirement.

For comparison, we briefly recall the one-more problem for a collection of trapdoor $(\mathcal{K}, \mathcal{F}, \mathcal{G}, \mathcal{R})$ from Definition 2.1.

Definition 2.14 (One-more trapdoor permutation problem).

Given the generation algorithm $\mathcal{G}$, the trapdoor permutation $f_{\text{pk}} \in \mathcal{F}$ and $\ell = \ell(k)$, for all PPT inverter algorithm $\mathcal{I}$ which does at most $\ell - 1 > 0$ queries to the inverter oracle f_{sk}^{-1} , we have that the probability

$$\Pr \left[\begin{array}{l} \mathcal{G}(1^k) \rightarrow (\text{sk}, \text{pk}); \\ \mathcal{R} \rightarrow (r_1, \dots, r_\ell); \\ \mathcal{I}^{f_{\text{sk}}^{-1}}(r_1, \dots, r_\ell, \text{pk}) \end{array} : \mathcal{I}^{f_{\text{sk}}^{-1}} \rightarrow (f_{\text{sk}}^{-1}(r_1), \dots, f_{\text{sk}}^{-1}(r_\ell)) \right] \leq \varepsilon_{OM}$$

where the probability is taken over the randomnesses of the algorithms $\mathcal{G}$, $\mathcal{R}$ and $\mathcal{I}^{f_{\text{sk}}^{-1}}$.

2.4.2 The One-More Variant in WPRM

We now formally state our definition for the one-more problem variant in the ρ -WPRM, i.e., WP-OM.

Definition 2.15 ($(\ell, \varepsilon_{WP-OM})$ -WP-OM). Let k be the security parameter and ρ be a GOOD_{OW} function. Given the generation algorithm $\mathcal{G}$, the trapdoor permutation $f_{\text{pk}} \in \mathcal{F}$ and $\ell = \ell(k)$, for all PPT inverter algorithm $\mathcal{I}$ which does at most $\ell - 1 > 0$ queries to the inverter oracle f_{sk}^{-1} , we have that the probability

$$\Pr \left[\begin{array}{l} \mathcal{G}(1^k) \rightarrow (\text{sk}, \text{pk}); \\ \mathcal{R} \rightarrow (r_1, \dots, r_\ell); \\ \text{GOOD}_{OW} \rightarrow \rho; \\ \mathcal{I}^{f_{\text{sk}}^{-1}, \rho}(r_1, \dots, r_\ell, \text{pk}) \end{array} : \mathcal{I}^{f_{\text{sk}}^{-1}, \rho} \rightarrow (f_{\text{sk}}^{-1}(\rho(r_1)), \dots, f_{\text{sk}}^{-1}(\rho(r_\ell))) \right] \leq \varepsilon_{WP-OM}$$

where the probability is taken over the randomnesses of the algorithms $\mathcal{G}$, $\mathcal{R}$ and $\mathcal{I}^{f_{\text{sk}}^{-1}, \rho}$.

The difference is that the inverter algorithm $\mathcal{I}^{f_{\text{sk}}^{-1}, \rho}$ receives the public key pk and $\ell(k)$ points in the domain, moreover it can query f_{sk}^{-1} and the GOOD_{OW} function ρ in a restricted number of times.

Intuitively, this definition is stronger than the standard one-wayness (when ℓ is fixed and equals to 1) of the trapdoor permutation. Consider, for example, that the algorithm $\mathcal{I}^{f_{\text{sk}}^{-1}, \rho}$ can choose which value it will try to invert among the ones it received from $\mathcal{R}$. This does not apply to the Definition 2.1, since, in there, $\mathcal{I}^{f_{\text{sk}}^{-1}, \rho}$ receives the single value it is expected to invert.

They were chosen by the arbitrary sampler algorithm $\mathcal{R}$, which samples them uniformly at random, and ρ .

Remark 2.16. As observed in [24] and [39], for $\ell_2 \geq \ell_1$, ℓ_2 -OM-RSA reduces to ℓ_1 -OM-RSA. The 1-OM-RSA is identical to the problem induced by Definition 2.1, i.e., the problem of inverting the one-way trapdoor permutation.

Remark 2.17. Note that the number of points is $\ell := \ell(k)$, that is it depends on the security parameter k .

2.5 The RSA-FDH in the WPRM

Here we give our security analysis of the RSA-FDH in WPRM.

For the next theorem, let $S = (\text{gen}, \text{sig}_{\text{sk}}, \text{ver}_{\text{pk}})$ be the RSA-FDH given by Section 2.2.6 and $\ell = \ell(k)$, where k is the security parameter.

Theorem 2.18. *Assume that the RSA cryptosystem, from Definition 2.2, is a collection of $(\ell, \varepsilon_{WP-OM})$ -WP-OM one-way trapdoor permutation as in Definition 2.15. Then, for all adversary $\mathcal{A}$ performing the EUF-CMA with at most q_s signature queries and q_h hash queries, such that $q_s + q_h \leq \ell - 1$, we have*

$$Adv_{S^{\mathcal{O}^\rho}}^{EUF-CMA}(\mathcal{A}) \leq \frac{q_s \cdot (q_s + q_h)}{2^k} + \varepsilon_{WP-OM},$$

where the probabilities are taken over the randomnesses of the adversary, the generation algorithm *gen* and WPRO.

Proof. Consider a challenger C which simulates the $\mathbf{Game}_{EUF-CMA}^{\mathcal{O}^\rho}$ to the adversary $\mathcal{A}$ as in Definition 2.4 with fresh randomness and security parameter k .

The challenger C simulates both the honest and the adversary interfaces as outlined in Definition 2.8. In other words, C keeps tables T and R , and samples uniformly random values r from the COINS set.

Assume for the sake of contradiction that $\mathcal{A}$ produces a valid forgery σ^* for some message m^* with non-negligible probability and denote by WIN the event that $\mathcal{A}$ outputs a valid forgery. The Definition 2.5 gives that

$$Adv_{S^{\mathcal{O}^\rho}}^{EUF-CMA}(\mathcal{A}) = \Pr[\mathsf{WIN}]$$

with probabilities taken over the randomness of the challenger C .

Let also QUERY be the event that $\mathcal{A}$ has queried m^* to the random oracle $\mathcal{O}^\rho$ and received z^* and r^* , while C defined $z^* = \rho(r^*)$ for a uniformly random value r^* as in Definition 2.8.

It is straightforward that

$$\begin{aligned} \Pr[\mathsf{WIN}] &= \Pr[\mathsf{WIN} \wedge \overline{\mathsf{QUERY}}] + \Pr[\mathsf{WIN} \wedge \mathsf{QUERY}] \\ &\leq \Pr[\mathsf{WIN} | \overline{\mathsf{QUERY}}] + \Pr[\mathsf{QUERY}] \end{aligned}$$

where the probabilities are taken from the randomnesses of challenger C and the generation algorithm *gen*.

We estimate the two probabilities $\Pr[\mathsf{WIN} | \overline{\mathsf{QUERY}}]$ and $\Pr[\mathsf{QUERY}]$ separately in the following two lemmas. The theorem follows.

Lemma 2.19. *Given a GOOD_{OW} function ρ , if the hash function h is modeled as the ρ -WPRO, then*

$$\Pr[\text{WIN} | \overline{\text{QUERY}}] \leq \frac{q_s \cdot (q_s + q_h)}{2^k}.$$

Proof. The event $\overline{\text{QUERY}}$ means that the adversary did not submitted a hash query on m^* . That is, the value $z^* = \rho(r^*)$ and r^* must have been defined in a signature query during the simulation of $\text{Game}_{\text{EUF-CMA}}^{\rho}$ given by C .

In order to understand why it must have been defined in a signature query, assume that the adversary $\mathcal{A}$ outputs (m^*, σ^*) without submitting m^* for a sign query nor a hash query. We show, next, that the probability that $\text{ver}(\text{pk}, m^*, \sigma^*) \rightarrow 1$ for such adversary is negligible. Therefore this adversary is often not considered proofs in the random oracle model.

Let SIGN be the event that m^* is submitted for a sign query in the simulation provided by C . Formally, we need to show that $\Pr[\text{ver}(\text{pk}, m^*, \sigma^*) \rightarrow 1 | \overline{\text{SIGN}} \wedge \overline{\text{QUERY}}]$ is negligible in k . First, observe that this probability is equal to

$$\Pr[\text{COINS} \rightarrow r' : y^* = \rho(r')]$$

where $y^* = (\sigma^*)^e$ is fixed by $\mathcal{A}$ by the time it outputs (m^*, σ^*) , and r' is uniformly chosen at random because $\overline{\text{SIGN}}$ and $\overline{\text{QUERY}}$, i.e., $(r', m^*) \notin R$, for R being the table for the WPRO simulation provided by the challenger C as in Definition 2.8. Since $(r', m^*) \notin R$, we are confident to state that by the time C checks the validity of σ^* , some r' is chosen when C runs $\text{ver}(\text{pk}, m^*, \sigma^*)$.

Recall that ρ is GOOD_{OW} , in particular, it is *regular*, therefore the size of the set of such values for r' is $\frac{|\text{COINS}|}{|\text{RNG}|}$. Moreover ρ is a $\{0, 1\}^c \rightarrow \{0, 1\}^k$ function, where k is the security parameter as defined by the RSA-FDH scheme. Hence the probability that the verification procedure picks the correct r' is upper bounded by

$$\Pr[\text{COINS} \rightarrow r' : y^* = \rho(r')] = \frac{1}{|\text{COINS}| \cdot \frac{1}{\frac{|\text{COINS}|}{|\text{RNG}|}}} = \frac{1}{2^k},$$

where r' is chosen uniformly at random. Therefore,

$$\Pr[\text{ver}(\text{pk}, m^*, \sigma^*) \rightarrow 1 | \overline{\text{SIGN}} \wedge \overline{\text{QUERY}}] \leq \frac{1}{2^k}.$$

The conclusion is that if the adversary outputs a *valid* forgery for m^* for the RSA-FDH, i.e., the verification algorithm returns 1, then we can assume that m^* was necessarily asked in a hash or a signature query.

Since by assumption **QUERY** does not happen during the simulation and the previous reasoning, we feel confident enough to assume that $z^* = \rho(r^*)$ and r^* must have been defined in a signature query. Then the challenger must have chosen, for some $m_i \neq m^*$ (otherwise it would not be a valid forgery), r_i such that $z^* = \rho(r^*) = \rho(r_i) = z_i$. This means

$$\Pr[\text{WIN}|\overline{\text{QUERY}}] = \sum_{i=1}^{q_s} \Pr[\text{COINS} \rightarrow r^* : \rho(r_i) = \rho(r^*) | \text{COINS} \rightarrow r_i].$$

Since by Definition 2.8, the values of r_i are chosen uniformly at random, the terms in the summation in the previous equation are upper bounded by all possible hashed values queried in q_s signatures and q_h hash queries,

$$\Pr[\text{COINS} \rightarrow r^* : \rho(r_i) = \rho(r^*) | \text{COINS} \rightarrow r_i] \leq \frac{q_s + q_h}{2^k}$$

for a fixed sign query i . Therefore, we have that

$$\Pr[\text{WIN}|\overline{\text{QUERY}}] \leq \frac{q_s \cdot (q_s + q_h)}{2^k}$$

for $0 < i \leq q_s$. □

Lemma 2.20. *Given a GOOD_{OW} function ρ and $q_s + q_h \leq \ell - 1$, if the hash function h is modeled as the ρ -WPRO and the RSA Cryptosystem has the property described in Definition 2.15, then*

$$\Pr[\text{QUERY}] \leq \varepsilon_{WP-OM}.$$

Proof. The adversary submitted a hash query on m^* , where m^* is the message which the signature was forged. Thus the value $z^* = \rho(r^*)$ and r^* must have been defined by the challenger C , i.e., there is a pair (r^*, m^*) in table **R** and a pair (z^*, m^*) in table **T** that C keeps during the simulation of $\mathbf{Game}_{\text{EUF-CMA}}^{\rho}$.

In order to bound the probability of the event **QUERY** to happen, assume that a direct hash query on m^* did happen, then we construct the inverter algorithm $\mathcal{I}_{\text{sk}}^{-1, \rho}$ as outlined in Definition 2.15. The inverter $\mathcal{I}_{\text{sk}}^{-1, \rho}$ is a black-box WPROM

reduction as described in Definition 2.13.

The construction of $\mathcal{I}_{\text{sk}}^{-1,\rho}$. Let $\{r_1, \dots, r_\ell\}$ be the input sampled by the sampling algorithm $\mathcal{R}$ which was received by the inverter algorithm $\mathcal{I}_{\text{sk}}^{-1,\rho}$ with pk . The inverter delivers pk to $\mathcal{A}$ and simulates the WPRO by keeping the tables T and R , as in Definition 2.8, and two lists, named USED and $\mathsf{INVERTED}$, both initially empty.

The algorithm $\mathcal{I}_{\text{sk}}^{-1,\rho}$ answers the hash and sign queries by using its ρ and f_{sk}^{-1} oracles, respectively, and through manipulation of $\mathsf{INVERTED}$ and USED , as follows:

- Hash query on x_i : choose a value r uniformly at random from $\{r_1, \dots, r_\ell\} \setminus \mathsf{USED}$ and add it to USED .
 1. forward r to the ρ oracle, and receive $\rho(r)$;
 2. add the pairs $(\rho(r), x_i)$ and (r, x_i) , respectively, to the tables T and R ;
 3. return r and $\rho(r)$ to the adversary;
- Sign query on m_i : If there is no x , such that $x = m_i$, for the pairs $(r, x) \in \mathsf{R}$ and $(\rho(r), x) \in \mathsf{T}$, then use the hash query procedure described earlier for m_i and run again the sign procedure. Otherwise, there is a pair $(\rho(r), m_i) \in \mathsf{T}$, then
 1. forward $\rho(r)$ to the oracle f_{sk}^{-1} , and receive $f_{\text{sk}}^{-1}(\rho(r))$, then;
 2. deliver $f_{\text{sk}}^{-1}(\rho(r))$ to the adversary as the signature σ_i for m_i ;
 3. add $f_{\text{sk}}^{-1}(\rho(r))$ to $\mathsf{INVERTED}$.

By assumption QUERY occurs. Note that, by definition, whenever the event QUERY happens the adversary $\mathcal{A}$ outputted a valid forgery σ^* for the message m^* . Then, it is known that $\sigma^* = f_{\text{sk}}^{-1}(\rho(r^*))$ for a pair $(\rho(r^*), m^*) \in \mathsf{T}$. Therefore, $\mathcal{I}_{\text{sk}}^{-1,\rho}$ adds $f_{\text{sk}}^{-1}(\rho(r^*))$ to $\mathsf{INVERTED}$.

For all remained value $r \in \{r_1, \dots, r_\ell\} \setminus \mathsf{USED}$, submit it to the oracle ρ and then to f_{sk}^{-1} and add the received value $f_{\text{sk}}^{-1}(\rho(r))$ to $\mathsf{INVERTED}$. Whereas, for all the values in USED which there is no inverted value in $\mathsf{INVERTED}$, submit to the f_{sk}^{-1} oracle. Thus, $\mathcal{I}_{\text{sk}}^{-1,\rho}$ outputs $\mathsf{INVERTED}$.

Analysis of $\mathcal{I}_{\text{sk}}^{f_{\text{sk}}^{-1}, \rho}$. Clearly, the running time of $\mathcal{I}_{\text{sk}}^{f_{\text{sk}}^{-1}, \rho}$ is polynomial-time in k . The key observation is that since h is modeled as a (non-programmable) random oracle, there are pairs $(r^*, m^*) \in \mathbf{R}$ and $(\rho(r^*), m^*) \in \mathbf{T}$. In order to see this, recall that it is very unlikely that the adversary can choose by itself a value $\rho(r^*)$ for a valid forgery when h is the random oracle. In other words, we may assume that the adversary received the correct (for a valid forgery) value $\rho(r^*)$ from the simulation.

In addition, since, by construction, all the values delivered in the hash queries are inputs of $\mathcal{I}_{\text{sk}}^{f_{\text{sk}}^{-1}, \rho}$ and $q_s + q_h \leq \ell - 1$, our inverter algorithm does at most $\ell - 1$ queries to its f_{sk}^{-1} oracle.

Hence, if the direct hash query on m^* happens and the adversary computes a forgery with non-negligible probability, that means

$$\Pr[\text{QUERY}] = \Pr[\mathcal{I}_{\text{sk}}^{f_{\text{sk}}^{-1}, \rho} \rightarrow (f_{\text{sk}}^{-1}(\rho(r_1)), \dots, f_{\text{sk}}^{-1}(\rho(r_\ell)))].$$

On the other hand, we assumed, from Definition 2.15, that for all PPT inverter algorithm $\mathcal{I}_{\text{sk}}^{f_{\text{sk}}^{-1}, \rho}$,

$$\Pr[\mathcal{I}_{\text{sk}}^{f_{\text{sk}}^{-1}, \rho} \rightarrow (f_{\text{sk}}^{-1}(\rho(r_1)), \dots, f_{\text{sk}}^{-1}(\rho(r_\ell)))] \leq \varepsilon_{WP-OM},$$

therefore we have

$$\Pr[\text{QUERY}] \leq \varepsilon_{WP-OM}$$

thereby giving the proof of the lemma. $\square$

2.6 A Tighter Security Reduction

For the next lemma, we consider the case when $q_s + q_h > \ell$. Let again ρ an GOOD_{OW} function, and now consider an arbitrary probability p . Our next construction for the inverter algorithm spreads the values from the one-more challenge within the hash values, in the hope that when the adversary output a forgery, it comes from $\{r_1, \dots, r_\ell\}$ since the adversary has no way to distinguish them.

The proof is more technical than the earlier one, therefore for reading convenience, let **INPUT** be the set $\{r_1, \dots, r_\ell\}$.

Lemma 2.21. *If the hash function h is modeled as the ρ -WPRO and the RSA Cryptosystem has the property described in Definition 2.15, and $q_s + q_h > \ell$ for $q_s \leq \ell - 1$, then*

$$\Pr[\text{QUERY}] \leq (1 - p)^{q_h} + \frac{\varepsilon_{WPRO}}{1 - q_s \cdot (q_h - (\ell - q_s)) \cdot (1 - p)} \cdot \frac{1}{1 - q_s \cdot (1 - (1 - p)^{q_h}) \cdot \frac{1-p}{p}}.$$

Proof. A direct hash query on m^* did happen, therefore the value $z^* = \rho(r^*)$ and r^* must have been defined by the challenger C , i.e., there is a pair (r^*, m^*) in table $\mathbf{R}$ and a pair (z^*, m^*) in table $\mathbf{T}$ that C keeps during the simulation of $\text{Game}_{\text{EUFCMA}}^{\mathcal{O}\rho}$.

If QUERY happens it means that the adversary could find the inverted value $f_{\text{sk}}^{-1}(\rho(r^*))$. Similarly, consider the next inverter algorithm $\mathcal{I}_{\text{sk}}^{f_{\text{sk}}^{-1}, \rho}$ as outlined in Definition 2.15.

The construction of $\mathcal{I}_{\text{sk}}^{f_{\text{sk}}^{-1}, \rho}$. The construction is similar to the one in the proof of Lemma 2.20, the differences are in the definitions of the simulations for the sign and the hash queries.

The inverter $\mathcal{I}_{\text{sk}}^{f_{\text{sk}}^{-1}, \rho}$ answers the hash and sign queries by using its ρ and f_{sk}^{-1} oracles, respectively, and through manipulation of $\text{INPUT} \setminus \text{USED}$, as follows:

- Hash query on x_i : If $\text{INPUT} \setminus \text{USED}$ is not empty, with arbitrary probability p , choose a value r uniformly at random from $\text{INPUT} \setminus \text{USED}$ and add it to USED . Otherwise, with probability $p - 1$ or $\text{INPUT} \setminus \text{USED}$ is empty, choose a uniformly random value r . Then,
 1. If r is in $\mathbf{R}$, i.e., $(r, x_i) \in \mathbf{R}$, then return r and $\rho(r)$, such that $(\rho(r), x_i) \in \mathbf{T}$, to the adversary.
 2. Otherwise, i.e., if $r \notin \mathbf{R}$, forward r to the ρ oracle, and receive $\rho(r)$.
 3. Add the pairs $(\rho(r), x_i)$ and (r, x_i) , respectively, to the tables $\mathbf{T}$ and $\mathbf{R}$;
 4. Return r and $\rho(r)$ to the adversary;
- Sign query on m_i :
If there is any $x = m_i$, such that $(r, x) \in \mathbf{R}$ and $(\rho(r), x) \in \mathbf{T}$. Then,

1. If $r \in \text{INPUT}$, then
 - (a) Forward $\rho(r)$ to the oracle f_{sk}^{-1} , and receive $f_{\text{sk}}^{-1}(\rho(r))$;
 - (b) Deliver $f_{\text{sk}}^{-1}(\rho(r))$ to the adversary as the signature σ_i for m_i .
- else
 - (a) Abort and let this event denote **Input-Abort**.

Else (if there is no x such that $x = m_i$)

1. If $\text{INPUT} \setminus \text{USED}$ is not empty
 - (a) Forward $\rho(r)$ to the oracle f_{sk}^{-1} , and receive $f_{\text{sk}}^{-1}(\rho(r))$;
 - (b) Deliver $f_{\text{sk}}^{-1}(\rho(r))$ to the adversary as the signature σ_i for m_i .
 - (c) add $f_{\text{sk}}^{-1}(\rho(r))$ to **INVERTED**.
- else
 - (a) Abort and let this event denote **Empty-Abort**.

By assumption, **QUERY** happens. As in the proof of Lemma 2.20 the adversary $\mathcal{A}$ outputted a valid forgery σ^* , and $\mathcal{I}_{\text{sk}}^{f_{\text{sk}}^{-1}, \rho}$ behaves in the same way as in the construction of the proof of Lemma 2.20.

Analysis of $\mathcal{I}_{\text{sk}}^{f_{\text{sk}}^{-1}, \rho}$. Clearly, the running time of $\mathcal{I}_{\text{sk}}^{f_{\text{sk}}^{-1}, \rho}$ is polynomial-time in k . The inverter algorithm succeeds if the following conditions are fulfilled:

- The value $r^* \in \text{INPUT}$ (therefore the event **QUERY** occurs); and
- $\mathcal{I}_{\text{sk}}^{f_{\text{sk}}^{-1}, \rho}$ does not abort.

This means

$$\begin{aligned} \Pr[\mathcal{I}_{\text{sk}}^{f_{\text{sk}}^{-1}, \rho} \text{ succeeds}] &= \Pr[\text{QUERY} | r^* \in \text{INPUT}] \cdot \Pr[r^* \in \text{INPUT}] \\ &\quad \cdot \Pr[\mathcal{I}_{\text{sk}}^{f_{\text{sk}}^{-1}, \rho} \text{ does not abort}]. \end{aligned} \tag{2.1}$$

On the other hand, it is straightforward to see that

$$\Pr[\text{QUERY}] \leq \Pr[r^* \notin \text{INPUT}] + \Pr[\text{QUERY} | r^* \in \text{INPUT}] \cdot \Pr[r^* \in \text{INPUT}].$$

From Equations 2.1 and 2.2, we have

$$\Pr[\mathcal{I}_{\text{sk}}^{f^{-1},\rho} \text{ succeeds}] \geq (\Pr[\text{QUERY}] - \Pr[r^* \notin \text{INPUT}]) \cdot \Pr[\mathcal{I}_{\text{sk}}^{f^{-1},\rho} \text{ does not abort}]. \quad (2.3)$$

The Definition 2.15 states that for all PPT inverter algorithm $\mathcal{I}_{\text{sk}}^{f^{-1},\rho}$,

$$\Pr[\mathcal{I}_{\text{sk}}^{f^{-1},\rho} \text{ succeeds}] \leq \varepsilon_{WP-OM}.$$

Therefore, Equation 2.3 gives

$$\Pr[\text{QUERY}] \leq \frac{\varepsilon_{WP-OM}}{\Pr[\mathcal{I}_{\text{sk}}^{f^{-1},\rho} \text{ does not abort}]} + \Pr[r^* \notin \text{INPUT}]. \quad (2.4)$$

At this point, we need to estimate the probabilities $\Pr[\mathcal{I}_{\text{sk}}^{f^{-1},\rho} \text{ does not abort}]$ and $\Pr[r^* \notin \text{INPUT}]$.

Estimation of $\Pr[\mathcal{I}_{\text{sk}}^{f^{-1},\rho} \text{ does not abort}]$. By our construction for $\mathcal{I}_{\text{sk}}^{f^{-1},\rho}$, note that

$$\Pr[\mathcal{I}_{\text{sk}}^{f^{-1},\rho} \text{ does not abort}] = \Pr[\overline{\text{Input-Abort}}] \cdot \Pr[\overline{\text{Empty-Abort}}].$$

We bound the probabilities separately. However, in both cases, consider without loss of generality that by the time of the j -th sign query there has been i hash queries already.

We start by $\Pr[\overline{\text{Empty-Abort}}]$.

First, recall that the inverter algorithm may have used at most j values through sign queries from its input values r , letting at least $\ell - j$ values left. Then the probability can be conditioned as follows

$$\begin{aligned} \Pr[\overline{\text{Empty-Abort}}] &= \Pr[\overline{\text{Empty-Abort}} | \ell - j > i] \cdot \Pr[\ell - j > i] \\ &\quad + \Pr[\overline{\text{Empty-Abort}} | \ell - j \leq i] \cdot \Pr[\ell - j \leq i]. \end{aligned}$$

The set $\text{INPUT} \setminus \text{USED}$ is not empty when $\ell - j > i$, therefore

$$\Pr[\text{Empty-Abort} | \ell - j > i] = 0.$$

Since

$$\Pr[\overline{\text{Empty-Abort}}] = 1 - \Pr[\text{Empty-Abort}]$$

then we know

$$\Pr[\overline{\text{Empty-Abort}}] > 1 - \Pr[\text{Empty-Abort} | \ell - j \leq i].$$

In other words, the probability $\Pr[\text{Empty-Abort} | \ell - j \leq i]$ is the same probability that, among the i hash queries, the inverter has sampled, from the set $\{r_1, \dots, r_\ell\} \setminus \text{USED}$, $\ell - j$ times (instead of picking a random value). The algorithm, by construction, chooses between values from the abovementioned set with arbitrary probability p and uniformly random values with probability $1 - p$, therefore

$$\Pr[\text{Empty-Abort} | \ell - j \leq i] = (1 - p).$$

Consequently, if $\ell - j \leq i \leq q_h$, then

$$\Pr[\text{Empty-Abort} | \ell - j \leq q_h] = \sum_{i=\ell-j}^{q_h} (1 - p) = (q_h - (\ell - j)) \cdot (1 - p)$$

for a fixed j signature queries.

The probability that the event **Empty-Abort** in any of the q_s signature queries is

$$\Pr[\text{Empty-Abort}] = \sum_{j=1}^{q_s} (q_h - (\ell - j)) \cdot (1 - p) = q_s \cdot (q_h - (\ell - q_s)) \cdot (1 - p),$$

and therefore

$$\Pr[\overline{\text{Empty - Abort}}] = 1 - q_s \cdot (q_h - (\ell - q_s)) \cdot (1 - p).$$

It remains to bound $\Pr[\overline{\text{Input-Abort}}]$.

Note that on the j -th signature query, the event **Input-Abort** happens with highest probability when, among all the i hash queries, the inverter $\mathcal{I}_{\text{sk}}^{-1, \rho}$ has not chosen any value from the set $\text{INPUT} \setminus \text{USED}$. This event happens with probability $(1-p)^i$ for a fixed number of hash queries i . Thus, we know that for $i \leq q_h$

$$\Pr[\text{Input-Abort} | \text{on the } j\text{-th sign query}] \leq \sum_{i=1}^{q_h} (1-p)^i. \quad (2.5)$$

When $1 \leq j \leq q_s$,

$$\begin{aligned} \Pr[\text{Input-Abort}] &\leq \sum_{j=1}^{q_s} \sum_{i=1}^{q_h} (1-p)^i \leq q_s \sum_{i=1}^{q_h} (1-p)^i, \\ &\leq q_s \cdot (1 - (1-p)^{q_h}) \cdot \frac{1-p}{p}. \end{aligned}$$

Consequently,

$$\Pr[\overline{\text{Input-Abort}}] \geq 1 - q_s \cdot (1 - (1-p)^{q_h}) \cdot \frac{1-p}{p}. \quad (2.6)$$

Estimation of $\Pr[r^* \notin \text{INPUT}]$. Similarly to the Equation 2.5, the worst case is when there was no query which $\mathcal{I}_{\text{sk}}^{-1, \rho}$ used values from its inputs, i.e., the set $\text{INPUT} \setminus \text{USED}$, among the q_h hash queries. Therefore we have the following upper-bound

$$\Pr[r^* \notin \text{INPUT}] \leq (1-p)^{q_h}. \quad (2.7)$$

The Equations 2.5, 2.6 and 2.7 plugged in the Equation 2.4 give that

$$\begin{aligned} \Pr[\text{QUERY}] &\leq (1-p)^{q_h} + \frac{\varepsilon_{WP-OM}}{1 - q_s \cdot (q_h - (\ell - q_s)) \cdot (1-p)} \\ &\quad \cdot \frac{1}{1 - q_s \cdot (1 - (1-p)^{q_h}) \cdot \frac{1-p}{p}}. \end{aligned}$$

Therefore we have the lemma. $\square$

Note that in the case of $p = 1$, we have the upper bound of the Lemma 2.20.

The Lemma 2.21 and the early mentioned Lemma 2.20 allow us to state a more general version of the Theorem 2.18.

Theorem 2.22. *Given an arbitrary probability $p < 1$, assume that the RSA cryptosystem, from Definition 2.2, is a collection of $(\ell, \varepsilon_{WP-OM})$ -WP-OM one-way trapdoor permutation as in Definition 2.15. Then, for all adversary $\mathcal{A}$ performing the EUF-CMA with at most $q_s < \ell - 1$ signature queries and $q_s + q_h > \ell$ hash queries, we have*

$$\begin{aligned} Adv_{SOP}^{EUF-CMA}(\mathcal{A}) \leq & \frac{q_s \cdot (q_s + q_h)}{2^k} + (1 - p)^{q_h} \\ & + \frac{\varepsilon_{WP-OM}}{1 - q_s \cdot (q_h - (\ell - q_s)) \cdot (1 - p)} \cdot \frac{1}{1 - q_s \cdot (1 - (1 - p)^{q_h}) \cdot \frac{1-p}{p}}, \end{aligned}$$

where the probabilities are taken over the randomnesses of the adversary, the generation algorithm gen and $WPRO$.

Proof. The proof follows from Lemma 2.19 and Lemma 2.21. □

Note that upper bound given by Theorem 2.22 does not give a proof of security since the term $(1 - p)^{q_h}$ is not negligible in general. However this can be set arbitrarily negligible by limiting the maximum number of signatures the signer party can issue. In other words, given the value ℓ , it is possible to derive the maximum number of signatures for which RSA-FDH remains EUF-CMA secure for a certain security parameter k .

We conclude by discussing the security regarding ℓ , k and q_s in the next section.

2.7 Security Analysis

The Theorem 2.22, means that if one assumes that the one-more RSA assumption holds for a fixed and concrete value ℓ , it is possible to derive a maximum number of signatures to be generated in order to remain the advantage negligible. As pointed by Coron in [5][Section 4], arbitrarily limiting the number of signatures is practical. We quote: “*This is practical significance since in the real-world applications, the*

number of hash calls is only limited by the computational power of the forger, whereas the number of signature queries can be deliberately limited.”

In other words, by setting $p = \frac{\ell - q_s}{q_h}$ and considering natural constraints, such as a high number of hash queries q_h and arbitrarily large value for $\ell - q_s$, the term $(1 - p)^{q_h}$ is not greater than $e^{-(\ell - q_s)}$, thus

$$\text{Adv}_{\mathcal{S}_{\mathcal{O}^\rho}}^{\text{EUF-CMA}}(\mathcal{A}) \leq \frac{q_s \cdot (q_s + q_h)}{2^k} + e^{-(\ell - q_s)} + \frac{\varepsilon_{\text{WP-OM}}}{1 + q_s},$$

thereby giving us a proof of security even in the non-programmable random oracle model, whenever $q_s < \ell - k \cdot \ln(2)$ signatures, where $\ln$ is the natural logarithm.

Note that in all cases, our results rely on the key (but not new) observation that the message from the forgery can be assumed to be hashed by the random oracle. Our result also shows that this property of the weaker variants of ROM seems to be even more crucial than the programmability of ROM.

2.8 Summary

We have shown that although the non-programmable ROM is a weaker model, the RSA-FDH is EUF-CMA secure under the assumption that the RSA cryptosystem is a collection of WP-OM one-way trapdoor permutation. This means that the programmability of ROM is not an essential property. Instead, our proof relies on the, apparently, more fundamental property: in the proof, we can always assume that the hash value, upon which the adversary has computed the forgery, was delivered by the random oracle.

Another observation is that the obtained lower bound for ℓ , i.e., $\ell > q_s + k \cdot \ln(2)$, is consistent to that in [16]. That is, the RSA-FDH cannot be proven secure in the non-programmable ROM assuming only the regular one-wayness ($\ell = 1$) property of the underlying trapdoor permutation.

There have been a few studies regarding the programmability of ROM, however, apparently, before the work of Larangeira and Tanaka [31], no such notion had been introduced to other models. In the next chapter, we review the work in [31] which formally introduced the notion of programmability in the Generic Group Model.

Chapter 3

Programmability in the Generic Group Model

In this chapter, we review basic definitions regarding the fundamental notions that the later sections rely on. In particular, we review the generic group oracle definitions and the Universally Composable Framework by Canetti [12]. We propose two different models for the Generic Group Model: Programmable GGM and Non-Programmable GGM. We show that the ElGamal Encryption scheme is secure in the Programmable GGM, while it is insecure in the Non-Programmable GGM due to the result by Nielsen [1]. In particular, we show that the ElGamal Encryption scheme is NINCE in the Programmable GGM.

3.1 The Generic Group Model Evolution

The ideas from the Generic Group Model came back from the works of Shoup [8] and Nechaev [9]. Initially to investigate the complexity of fundamental problems in cryptography, such as the discrete logarithm problem, as well as, the relationship between breking RSA and factoring integers [40, 41].

Maurer and Wolf in [42–44] investigated the relation between breaking the Diffie-Hellman protocol, based on the Diffie-Hellman problem [45], and the discrete logarithm problem. Furthermore den Boer [46] showed similarities between the problems in certain cases. Boneh and Lipton [47] also investigated the black-box field which is a similar notion to generic group. Maurer’s formulation of GGM is,

in fact, different from Shoup's, however Jager and Schwenk, in [20], have showed that both models are equivalent.

More recently models based on the GGM have been used to argue about the hardness of newly introduced computational problems [23, 48] as well as the security of cryptographic schemes [49–53].

We start by reviewing the Universally Composable Framework.

3.2 Preliminaries

We start by reviewing the universal composable framework, in particular the hybrid extension of the framework. We refer the reader to [12] for more details.

3.2.1 The Universal Composable Framework

In the framework all the parties are probabilistic polynomial time (PPT) interactive Turing machines¹. Given the sender and receiver parties, SEN and REC , the protocol $\pi = (SEN, REC)$ performs a REAL execution of the protocol when SEN and REC run the code of π in the presence of the adversary A . Moreover, A can see and schedule the messages between the parties. On the other hand, there is the IDEAL execution which is defined by the functionality F , the simulator S , and the parties $\widetilde{SEN}$ and $\widetilde{REC}$. The parties in the IDEAL execution only deliver its inputs to F and passes the outputs of F without performing any computation. Both executions are driven by the environment Z , which plays the role of the distinguisher and controls the inputs of the parties as well as the actions of A .

The hybrid model. In addition with the previous model, the hybrid model is the extension where all parties have secure access to a special machine G named also functionality. At the beginning, the environment receives the security parameter k and an auxiliary input $z \in \{0, 1\}^*$. Let $\text{HYB}_{\pi, A, Z}^G(k, z)$ be the random variable

¹Which are Turing machines equipped with three extra tapes: (1) The input tape, (2) the communication tape, and (3) the subroutine output tape.

describing the output of Z in the hybrid execution. Whereas $\text{HYB}_{\pi, S, Z}^G(k, z)$ is for the ideal execution. Let $\text{HYB}_{\pi, S, Z}^G(k, z) = \text{IDEAL}_{F, S, Z}(k, z)$.

It is said that π *securely realizes* F in the G -hybrid model, when for any adversary A there exists an ideal adversary S such that for any environment Z ,

$$| \Pr[\text{IDEAL}_{F, S, Z}(k, z) = 1] - \Pr[\text{HYB}_{\pi, A, Z}^G(k, z) = 1] | < \text{negl}(k).$$

3.2.2 The Non-Committing Encryption Protocol

We define the non-committing protocol π by adding an identification header in the ciphertexts. That is three unique strings id_m , id_{SEN} and id_{REC} . Let π be defined as follows:

send: On input $(\text{send}, id_m, id_{REC}, m)$, SEN computes

$$c = \text{enc}^G(\text{pk}, id_m || id_{SEN} || id_{REC} || m)$$

and sends (id_m, c) to REC .

receive: If REC receives (id_m, c) , it computes

$$\text{dec}^G(\text{sk}, c) = id_m || id_{SEN} || id_{REC} || m$$

and outputs $(\text{receive}, id_m, id_{SEN}, m)$.

Let $l_h(k)$ be the size in bits of $id_m || id_{SEN} || id_{REC}$, while $l_m(k)$ is the size of m in bits regarding the security parameter k . The negative result from [1] is related to non-interactive non-committing encryption (NINCE) schemes.

Non-interactive schemes are those which does not allow any more exchange of information after a preprocessing phase. Allowing the parties to send and receive an unbound amount of information. While the non-committing schemes are modeled through the functionality F_{ncc} , and which we now review.

3.2.3 The Non-Committing Encryption Functionality F_{nce}

The functionality can be described for the protocol $\tilde{\pi} = (\widetilde{SEN}, \widetilde{REC})$ in the IDEAL execution as follows:

1. F_{nce} sends $(\text{send}, id_m, id_{SEN}, id_{REC}, l_h(k) + l_m(k))$ to S , whenever $\widetilde{SEN}$ inputs $(\text{send}, id_m, id_{REC}, m)$ to F_{nce} .
2. F_{nce} waits S schedule the delivery of $(\text{receive}, id_m, id_{SEN}, m)$ from F_{nce} to $\widetilde{REC}$.
3. When S allows it, F_{nce} sends $(\text{receive}, id_m, id_{SEN}, m)$ to $\widetilde{REC}$.

Now we review the formalization of the GGM and the crucial lemma, the Schwartz Lemma [32], for the model.

3.2.4 The Schwartz Lemma

Lemma 3.1 (Schwartz Lemma [8, 32]). *Let p be prime and $L(X_1, \dots, X_\ell)$ an arbitrary non-zero polynomial on the random variables $X_1, \dots, X_\ell \in \mathbb{Z}_p$ with total degree d for $\ell \in \mathbb{N}$. Then $\Pr[L(x_1, \dots, x_\ell) = 0] \leq \frac{d}{p}$ for a uniformly random assignment $X_i \leftarrow x_i \in \mathbb{Z}_p$, $1 \leq i \leq \ell$.*

The early defined lemma is used combined with the definition of generic group algorithms that we review now.

3.2.5 The Generic Group Algorithms

A generic group algorithm A is allowed to submit operation and equality queries to the generic group operation oracle G . The queries are tuples $(i, j, \circ)$ for $\circ \in \{=, \times\}$, where $\times$ is the group operation.

Definition 3.2 (Generic Group Oracle G [8, 11]). The oracle G initializes by receiving a value $x \in \mathbb{Z}_p^*$ and keeping a list of polynomials L_r , $1 \leq r \leq q(k)$ associated with every submitted operation $(i_r, j_r, \circ_r)$, with additional convention that $L_{-1} := 1$ and $L_0 := x$. The adversary submits queries $(i_r, j_r, \circ_r)$ and G runs the **operation** procedure:

operation($i_r, j_r, \circ_r$):

1. It takes a tuple $(i_r, j_r, \circ_r) \in \{-1, \dots, r-1\} \times \{-1, \dots, r-1\} \times \{=, \times\}$
2. If $\circ_r \neq \{=\}$ then
 - If $\exists r' \neq r : L_{r'} = L_{i_r} \times L_{j_r}$ then **return** r'
 - Else
 - $L_r := L_{i_r} \times L_{j_r}$
 - return** r
- Else
 - If $L_{i_r} = L_{j_r}$ then **return** *true*
 - Else **return** *false*

TABLE 3.1: Definition of the routine for the generic group operation oracle $\mathbf{G}$.

The complexity of $\mathbf{A}$ is measured as the number of oracle queries.

3.3 The Programmable and Non-Programmable GGM

Here we introduce the *programmable* and *non-programmable* models of GRM and GGM.

As in [1], the technical difference is as follows: In the programmable setting, $\mathbf{Z}$ does not have access to the generic functionality, whereas it does in the non-programmable setting. Moreover, let the functionality $\mathbf{G}$ execute the routines described in Definition 3.2 for GGM.

3.3.1 The Programmable GGM

This is the regular GGM. All parties of protocol π have access to the generic group oracle $\mathbf{G}$, and we say that π securely realizes some functionality $\mathbf{F}$ in the programmable GGM, if π *securely realizes $\mathbf{F}$ in the $\mathbf{G}$ -hybrid model*.

3.3.2 The Non-programmable GGM

In this variant, the environment Z has access to the generic operation oracle G . Thus, any computation made by the parties of π can be carried out by the environment itself. It is said that π *securely realizes some functionality F in the non-programmable GGM*.

3.4 The ElGamal Encryption Scheme in the Programmable GGM

Here we review the ElGamal encryption scheme and study its security in the programmable GGM.

3.4.1 The ElGamal Encryption Scheme

Let $[m_{\mathbb{G}}]$ be the pointer to the group element $m_{\mathbb{G}} \in \mathbb{G}$ in the simulation of G , where $\mathbb{G}$ is a group of k -bit long prime order p generated by g , both outputted by a generation algorithm gen . There is no need to specify the size of the message $|m_{\mathbb{G}}|$ because it is known that in the ElGamal's case the message is encoded as a group member. Therefore, we assume the existence of an encoding function $e : \{0, 1\}^{\log_2 p} \rightarrow \mathbb{G}$.

Note that in GGM the computation $e^G(id_m || id_{SEN} || id_{REC} || m)$ gives the pointer $[m_{\mathbb{G}}]$, where $1 < m_{\mathbb{G}} \leq p - 1$. Moreover, $m_{\mathbb{G}}$ does not depend on the secret key, and $[m_{\mathbb{G}}]$ must have been delivered by G .

The scheme is described in Table 3.2.

3.4.2 The Security of the ElGamal Encryption Scheme in the Programmable GGM

We show that our protocol π based on the ElGamal encryption scheme is non-committing in the programmable GGM in the presence of the adversary who corrupts the receiver and the sender and expose the secret key sk .

$gen(1^k)$	$enc(pk, m_{\mathbb{G}})$	$dec(sk, c)$
$(g, p) \leftarrow \text{gen}(1^k)$ $x \xleftarrow{U} \mathbb{Z}_p^*$ $pk \leftarrow (g, g^x)$ $sk \leftarrow (g, x)$ $\text{return}(pk, sk)$	$(g, g^x) \leftarrow pk$ $r \xleftarrow{U} \mathbb{Z}_p^*$ $c_1 \leftarrow g^r$ $c_2 \leftarrow (g^x)^r \cdot m_{\mathbb{G}}$ $c \leftarrow (c_1, c_2)$ $\text{return } c$	$(g, x) \leftarrow sk$ $y \leftarrow (c_1)^x$ $m_{\mathbb{G}} \leftarrow c_2 \cdot y^{-1}$ $\text{return } m_{\mathbb{G}}$

TABLE 3.2: The ElGamal encryption scheme.

Theorem 3.3. *Let k be the security parameter and p a k -bit long prime. Given the protocol $\pi = (REC, SEN)$ which allows every party to send $n(k)$ messages, there is a simulator S such that for every environment Z , and every adversary A which can corrupt REC and SEN and makes $q(k)$ queries to the generic group operation oracle G , then*

$$|\Pr[IDEAL_{F, S^G, Z^G}(k, z) = 1] - \Pr[HYB_{\pi, A, Z}^G(k, z) = 1]| \leq \frac{q(k)^2 \cdot (n(k) + 1)}{p} + \text{negl}(k),$$

for the probabilities taken over the random choices of the environment Z in the programmable GGM.

Proof.

Overview. We construct a simulator S which simulates the ideal execution for the adversary A including the generic group operation oracle G_{PRO} . In comparison to G from Definition 3.2, G_{PRO} does not have a fixed challenge value x . Instead, it leaves as random variable X unless the adversary corrupts the receiver. Upon this event, G_{PRO} fixes X with a random x . We show that the probability of Z in distinguishing the hybrid and the ideal executions.

First, consider that the correctness of the scheme gives

$$\Pr[HYB_{\pi, A, Z}^G(k, z) = 1] = 1 - \text{negl}(k),$$

for the random choices of the environment Z .

We now construct a simulator S for the ideal execution $IDEAL_{F, S^G, Z^G}(k, z)$. The simulator S must handle two situations: (1) when F_{nce} sends $(\text{send}, id_m, id_{SEN}, id_{REC})$ and (2) when SEN sends $(id_m, [c])$ to REC in the simulation of the hybrid execution.

Send: (1) S must deliver to A $(id_m, [c])$ whenever it receives the tuple

$$(\mathbf{send}, id_m, id_{SEN}, id_{REC})$$

from F_{nce} .

- If $\widetilde{REC}$ is corrupted:
 - S allows $(\mathbf{receive}, id_m, id_{SEN}, m_{\mathbb{G}})$ to be sent to $\widetilde{REC}$.
 - Since $\widetilde{REC}$ is corrupted, S learns $m_{\mathbb{G}}$. Then it computes

$$c = enc^{G_{PRO}}(\mathbf{pk}, [m_{\mathbb{G}}])$$

and delivers $(id_m, [c])$ to A .

- If $\widetilde{REC}$ is honest:
 - S does not know $[m_{\mathbb{G}}]$, therefore it adds to its simulation of G_{PRO} (as in Definition 3.2) two random variables L_{c_1} and L_{c_2} both sampled from $\mathbb{Z}_p^*$.
 - S gives A the pointers $[L_{c_1}]$ and $[L_{c_2}]$, as it would normally do if it knew the values for L_{c_1} and L_{c_2} .
 - S continues simulating G_{PRO} , now with X , L_{c_1} and L_{c_2} as unknowns.

If, at a later moment, SEN or REC is corrupted:

- If SEN was corrupted
 - * S corrupts $\widetilde{SEN}$ and learns $[m_{\mathbb{G}}]$, and therefore $m_{\mathbb{G}}$.
 - * S partially commits to the ciphertext by choosing uniformly at random $1 \leq r < p - 1$. If there is some i such that $L_i = r$, then S sets $L_{c_1} \leftarrow L_i$. Otherwise S sets $L_{c_1} \leftarrow r$. Therefore the partially committed ciphertext $[c] = ([L_{c_1}], [L_{c_2}])$ is

$$L_{c_1} \leftarrow r, L_{c_2} \leftarrow r \cdot X + m_{\mathbb{G}},$$

with X still as an unknown.

- * S delivers r and $[m_{\mathbb{G}}]$ to A as its internal view of SEN .
- If REC was corrupted
 - * S corrupts $\widetilde{REC}$ and learns $[m_{\mathbb{G}}]$ and therefore $m_{\mathbb{G}}$.

- * S chooses the secret key x uniformly at random from $\mathbb{Z}_p^*$.
- * Then S commits to the correct ciphertext by choosing uniformly at random $1 \leq r < p - 1$ and fixing $X \leftarrow x$ and $m_{\mathbb{G}}$,

$$L_{c_1} \leftarrow r, L_{c_2} \leftarrow r \cdot x + m_{\mathbb{G}}.$$

- * S aborts if the simulation was not perfect. Let this event be denoted by **abort**.
- * If S did not abort, it delivers x and $[m_{\mathbb{G}}]$ to A as its internal view of REC .

Receive: (2) S must make F_{ncc} output $(\text{receive}, id_m, id_{SEN}, [m_{\mathbb{G}}])$ to $\widetilde{REC}$, whenever SEN sends $(id_m, [c])$ to REC .

- If SEN is honest
 Then $(id_m, [c])$ was sent by S in response to what happen in the ideal world. Then, S must have received $(\text{send}, id_m, id_{SEN}, id_{REC})$ from F_{ncc} . Therefore, S allows $(\text{receive}, id_m, id_{SEN}, [m_{\mathbb{G}}])$ to be sent to $\widetilde{REC}$ in the ideal execution.
- If SEN is corrupted
 - S looks for $[L_{c_1}]$ and $[L_{c_2}]$ such that $[c] = ([L_{c_1}], [L_{c_2}])$ and it must find two polynomials L_{c_1} and L_{c_2} on X .
 - S creates a pointer $[m_{\mathbb{G}}]$ associated with the polynomial $L_{c_2} - L_{c_1} \cdot X$ (as in the decrypting algorithm). Note that the addition of $[m_{\mathbb{G}}]$ adds one degree in the polynomials carried by G_{PRO} every time SEN sends a new pair $(id_m, [c])$. Therefore, the degree can be at most $n(k) + 1$, where $n(k)$ is the number of messages a single party sends.
 - S inputs $(id_m, id_{REC}, [m_{\mathbb{G}}])$ in $\widetilde{SEN}$, since $\widetilde{SEN}$ is also corrupted.
 - S allows F_{ncc} to send $(\text{receive}, id_m, id_{SEN}, [m_{\mathbb{G}}])$ to $\widetilde{REC}$.

It remains to analyze the abort probability of S . If the simulation does not abort the views of the hybrid and ideal executions are indistinguishable. Therefore, let us assume for the sake of contradiction that the simulation aborts on event **abort**. This event occurs when some polynomial in the simulation are different, however

they evaluate to the same value after fixing L_{c_1} and L_{c_2} with x chosen uniformly at random.

Recall that on the event **abort** the simulated oracle $\mathbf{G}_{PRO}$ has at most $q(k)$ polynomials: $L_1, \dots, L_{q(k)}$. Lemma 3.1 gives that the polynomial defined as $L_i - L_j$, for $1 \leq i, j \leq q(k)$, $i \neq j$, has $\Pr[(L_i - L_j)(x) = 0] \leq \frac{n(k)+1}{p}$ for uniformly random choices of $x \leftarrow \mathbb{Z}_p$. Since the total degree of $(L_i - L_j)$ can be at most $n(k) + 1$, therefore,

$$\begin{aligned}
 \Pr[\text{abort}] &= \Pr_{\substack{1 \leq i, j \leq q(k) \\ i \neq j}} [(L_i - L_j)(x) = 0 | x \xleftarrow{U} \mathbb{Z}_p] \\
 &= q(k) \cdot (q(k) - 1) \cdot \Pr[L(x) = 0 | x \xleftarrow{U} \mathbb{Z}_p] \\
 &\stackrel{\text{Lemma 3.1}}{\leq} q(k) \cdot (q(k) - 1) \cdot \frac{n(k) + 1}{p} \\
 &\leq \frac{q(k)^2 \cdot (n(k) + 1)}{p},
 \end{aligned}$$

where L is a polynomial with total degree $n(k) + 1$ and probabilities are taken over the choices of x . That is

$$\Pr[\text{IDEAL}_{\mathbb{F}, S^G, \mathbb{Z}^G}(k, z) = 1] \leq 1 - \frac{q(k)^2 \cdot (n(k) + 1)}{p},$$

thereby giving the theorem. $\square$

3.5 Summary

The proof paradigm in this chapter is different from the early one. Here, we have considered the simulation paradigm in comparison to the reduction-centric fashion of Chapter 2. Similarly to [1], we have used the hybrid extension to the UC Framework to define our novel models: the programmable and non-programmable GGM.

We have shown that the non-programmable GGM is indeed a weaker model by showing that the non-committing encryption schemes exists in the programmable GGM, since Nielsen's result means that non-committing encryptions schemes does not exist.

In particular, we have shown that the ElGamal encryption scheme is NINCE scheme regarding adaptive adversaries and corruption of both senders and receivers and fully exposure of the encryption key.

It is natural to ask whether such a result extends to a more general model. In the next chapter we explore this question by investigating the generalization of the GGM: the Generic Ring Model.

Chapter 4

Programmability in the Generic Ring Model

In this chapter, likewise the last chapter, we propose two models for the Generic Ring Model: Programmable GRM and Non-Programmable GRM. Furthermore, we show that the Goldwasser-Micali Encryption scheme is secure in the Programmable GRM, while it is insecure in the Non-Programmable GRM due to the result by Nielsen [1]. In particular, we show that the Goldwasser-Micali Encryption scheme is NINCE in the Programmable GRM when considering only receiver corruption. This result does not rule out the existence of full NINCE scheme (considering also sender corruptions), but is enough to circumvent the result in [1].

We start by reviewing the evolution of GRM.

4.1 The Generic Ring Model Evolution

The Generic Ring Model is based on the notion of *straight line programs* (SLP) (formally introduced later) which was first suggested by Boneh and Venkatesan [21] as a tool to investigate the relation between breaking RSA and the factoring problem, which is a long standing question in Cryptography.

In the realm of provable security, understand the relation between assumed hard problems and functions or trapdoors is of prime importance. Furthermore, the

RSA family of trapdoor permutations is at the heart of several cryptosystems in industry. No wonder this sort of investigation draws much attention of the community.

The result in [21] was followed by a myriad of works [40, 41, 54, 55] each of them proposing a different, and often more general, formulation for the model. All of these works have in common the goal of investigating the relation between the already problems of factoring and breaking RSA, and, so far, the matter is not settled. While Brown [54] and Leander and Rupp [41] presented arguments favouring the assumption that breaking RSA is equivalent to factoring, Boneh and Venkatesan [21] and Joux et al [55] present arguments supporting otherwise.

Maurer proposed a model to embrace these formulations in [11] as well as those for GGM, and was followed by Aggarwal and Maurer [10]. While the former presented results regarding GGM, the later extended all the already mentioned works regarding models of computation and introduced the so called Generic RSA assumption.

The ideas in [10][Proposition 1 and Lemmas 6, 7 and 8] appeared again later in [22] where Jager and Schwenk showed that another computational problem, computing the Jacobi symbol, and factoring are equivalent in GRM. Although, it is not surprising, the meaning of their result is that the GRM is indeed a stronger model, because it is known that it is possible to efficiently compute the Jacobi Symbol with non-generic algorithms (algorithms that cannot be cast into the GRM).

The research in the GRM has been evolving and the formulation in [22] is the simplest and more general we could find in the literature up to this point. Therefore, we start this chapter by reviewing the definitions from [22] and [10].

4.2 Preliminaries

4.2.1 Straight Line Programs

From here, assume that $N = p \cdot q$ for primes p and q , where $\phi(N) = |\mathbb{Z}_N^*| = (p - 1) \cdot (q - 1)$ is the Euler totient function.

We adapted the next definitions and lemmas from [22, Lemmas 2, 3 (Appendix C) and 4 (Appendix D), respectively] and ideas from [10]. The main change is that now we consider the tuples X and Y defined as $X = (x_1, \dots, x_\ell)$ and $Y = (y_1, \dots, y_\ell)$ where $\ell \in \mathbb{N}$ and x_i and y_i are sampled independently. The originals are defined over single values. We skip the proofs of the lemmas, therefore we refer the reader both works for more details.

Definition 4.1 (Straight Line Programs (SLP) [21, 22]). A straight line program P of length m over $\mathbb{Z}_N$ is a sequence of tuples $P = ((i_1, j_1, \circ_1), \dots, (i_m, j_m, \circ_m))$, where $-\ell \leq i_k, j_k < k$ and $\circ_r \in \{+, -, \times, /\}$ for $r \in \{1, \dots, m\}$, with the additional convention $P_0(X) = 1$, $P_{-1}(X) = x_1, \dots, P_{-\ell}(X) = x_\ell$ for all $x_i \in \mathbb{Z}_N$, where $1 \leq i \leq \ell$.

The triple $(i, j, \circ) \in P$ is an *SLP-step*. We denote by P_k the SLP defined by the first k SLP-steps of P and $P_k \sqsubseteq P$ to denote that the SLP P_k means the first k SLP-steps of P .

The output $P(X)$ of an SLP P is computed as follows.

1. For $1 \leq i \leq \ell$, initialize $L_{-i} := x_i \in \mathbb{Z}_N$ and $L_0 := 1$.
2. For k from 1 to m do:
 - If $\circ_k = /$ and $L_{j_k} \notin \mathbb{Z}_N^*$ then return $\perp$;
 - else set $L_k := L_{i_k} \circ_k L_{j_k}$.
3. Return $P(x) = L_m$.

4.2.2 Generic Ring Algorithms

A generic ring algorithm A is allowed to submit SLP-steps $(i, j, \circ)$ to the generic ring operation oracle R .

Definition 4.2 (Generic Ring Oracle R [22]). The oracle R is initialized by keeping a SLP P initially empty except for $P_{-1}(x) = x$ and $P_0(x) = 1$ and the received value $x \in \mathbb{Z}_N$. The checking procedures are done by implementing the routines **test** and **equal**. The algorithm A submits SLP-steps $(i, j, \circ)$ to perform computation. For every tuple $(i, j, \circ)$, R executes the procedure **operation**:

test ($j, \circ$): 1. It takes a tuple $(j, \circ) \in \{-1, \dots, P \} \times \{+, -, \times, /, =\}$ 2. If $\circ = /$ and $P_j(x) \notin \mathbb{Z}_N^*$ then Return false Else Return true	equal (j, i): 1. It takes a tuple $(j, i) \in \{-1, \dots, P \} \times \{-1, \dots, P \}$ 2. If $P_i(x) \equiv P_j(x) \pmod n$ then Return true Else Return false
---	--

operation($i, j, \circ$):
 1. It takes a tuple $(i, j, \circ)$
 2. If $\circ \neq \{=\}$ then
 If **test**($j, \circ$) $\rightarrow$ false then
 Return $\perp$
 Else add $(i, j, \circ)$ to P
 Else
 Return **equal**(i, j)

TABLE 4.1: Definition of the routine for the generic ring operation oracle R.

The complexity of **A** is measured as the number of oracle queries.

Remark 4.3. In order to program the oracles we need an additional convention. For a member x of the ring or group, $[x]$ means the pointer which **G**, or **R**, give to **A**, so the adversary can carry out the computation.

Lemma 4.4 ([31]). *Suppose there exist a straight line program P such that for $X, X' \in (\mathbb{Z}_N^*)^\ell$ it holds that $P(X') \neq \perp$ and $P(X) = \perp$. Then there exists $P_j \sqsubseteq P$ such that $P_j(X') \in \mathbb{Z}_N^*$ and $P_j(X) \notin \mathbb{Z}_N^*$.*

Lemma 4.5 ([31]). *For every SLP P ,*

$$\begin{aligned} \Pr[\gcd(P(Y), N) \notin \{1, N\} \mid Y \xleftarrow{U} (\mathbb{Z}_N^*)^\ell] \\ \geq \Pr[P(X) \notin \mathbb{Z}_N^* \text{ and } P(X') \in \mathbb{Z}_N^* \mid X, X' \xleftarrow{U} (\mathbb{Z}_N^*)^\ell]. \end{aligned}$$

Lemma 4.6 ([31]). *For every two SLP P and Q ,*

$$\begin{aligned} \Pr[\gcd(P(Y) - Q(Y), N) \notin \{1, N\} \mid Y \xleftarrow{U} (\mathbb{Z}_N^*)^\ell] \\ \geq \Pr[P(X) \neq_N Q(X) \text{ and } P(X') \equiv_N Q(X') \mid X, X' \xleftarrow{U} (\mathbb{Z}_N^*)^\ell]. \end{aligned}$$

Next, we introduce our models.

4.3 The Programmable and Non-Programmable GRM

Our models are based on the simulation paradigm and are defined using the hybrid model of UC.

4.3.1 Programmable-GRM

This is the regular GRM. All parties of protocol π have access to the generic group oracle R , and we say that π securely realizes some functionality F in the programmable GRM, if π *securely realizes F in the R -hybrid model*.

4.3.2 Non-programmable-GRM

In this variant, the environment Z has access to the generic operation oracle R . Thus, any computation made by the parties of π can be carried out by the environment itself. It is said that π *securely realizes some functionality F in the non-programmable-GRM*.

As in the previous Section 3.3 and in [1], the technical difference is as follows: In the programmable setting, Z does not have access to the generic functionality, whereas it does in the non-programmable setting. Moreover, let the functionality R execute the routines described in Definition 4.2 for GRM.

4.4 The Goldwasser-Micali Encryption Scheme in the Programmable GRM

Here we review the Goldwasser-Micali (GM) encryption scheme and study its security in the programmable GRM.

4.4.1 The Goldwasser-Micali Encryption Scheme

First, let the algorithm **gen** denote the generation algorithm which gives two large prime numbers p and q , and their product N . Let QR_N denote the set of the quadratic residue modulo N , while QNR_N denote the set of non-residues. Moreover, let QNR_N^+ denote the set of the quadratic non-residues which the Jacobi symbol modulo N gives $+1$. That is, given $x \in \text{QNR}_N$, $J_N(x) = +1$.

For the scheme, consider that m_i , $1 \leq i \leq l_m(k)$, are the bits of the message m , while $c_i \in \mathbb{Z}_N$ and $r_i \in \{0, 1\}^{l_r(k)}$ are respectively the ciphertexts and the randomness for each bit.

The scheme is described in Table 4.2.

$gen(1^k)$	$enc(pk, m)$	$dec(sk, c)$
$(p, q, N) \leftarrow \text{gen}(1^k)$ $x \xleftarrow{U} \text{QNR}_N^+$ $pk \leftarrow (N, x)$ $sk \leftarrow (p, q)$ return (pk, sk)	$(N, x) \leftarrow pk$ For $i = 1$ to $l_m(k)$ $r_i \xleftarrow{U} \mathbb{Z}_N^*$ $c_i \leftarrow r_i^2 \cdot x^{m_i}$ return c	$(p, q) \leftarrow sk$ For $i = 1$ to $l_m(k)$ If $J_p(c_i) = +1$ and $J_q(c_i) = +1$ $m_i \leftarrow 0$ Else $m_i \leftarrow 1$ return m

TABLE 4.2: The Goldwasser-Micali encryption scheme.

Remark 4.7. The ciphertext $c = (c_1, \dots, c_{l_m(k)})$ of a message $m \in \{0, 1\}^{l_m(k)}$ is composed by c_i which are members of the ring. Therefore they are represented in the model as SLP-steps. The public key pk is the SLP's $P_0(x)$ and $P_{-1}(x)$ as in Definition 4.1.

4.4.2 The Security of the GM Encryption Scheme in the Programmable GRM

We show that our protocol π for GM is non-committing in the programmable GRM even in the presence of the adversary who corrupts the receiver and expose the secret key sk .

Theorem 4.8. *Let k be the security parameter. Let $\pi = (REC, SEN)$ be the protocol, then there is a simulator S such that for every environment Z , and every adversary A which can corrupt the receiver party REC and makes at most $q(k)$ queries to the generic ring operation oracle R , then*

$$\begin{aligned} |\Pr[IDEAL_{F, S^G, Z^G}(k, z) = 1] - \Pr[HYB_{\pi, A, Z}^G(k, z) = 1]| \\ \leq 2q(k) \cdot (q(k)^2 + 5q(k) + 3) \cdot \varepsilon_{FACT} \end{aligned}$$

and probability ε_{FACT} of finding a factor of N in the programmable GRM.

Proof.

Overview. We construct a simulator S which simultaneously tries to factor the modulus N and simulates R and interacts with the ideal execution with F_{nce} . The simulator generates SLP-steps $P^{(m_i)}(x)$ as the ciphertext, and provides A with the pointers $[P^{(m_i)}(x)]$ to them until learning the message m .

The simulator provides A with the simulation of the operation oracle R_{PRO} when simulating the ciphertexts. After receiving m , S commits to the given SLP-steps with the message m , and instead of R_{PRO} simulates again R . We show that the probability of S in simulating the ideal execution is upper bounded by the probability of factoring N .

A challenger runs $gen(1^k) \rightarrow (p, q, N)$, and gives S the value N . The simulator S is allowed to give up the value, and then receives (p, q) and loses the game. It wins if outputs a value y such that $\gcd(y, N) \neq \{N, 1\}$. Denote the success probability in the game ε_{FACT} .

We need to define a simulated oracle R_{PRO} . The simulated oracle R_{PRO} is the same as R , however instead of implementing **test** and **equal**, R_{PRO} implements **testpro** and **equalpro**, which are defined in Table 4.3.

We present the construction for S . Initially it receives the modulus N and picks $x \xleftarrow{U} \mathbf{QNR}_N^+$ and runs R as in Definition 4.2 with $P_{-1}(x) = x$ and $P_0(x) = 1$. The simulator must handle two situations: (1) when F_{nce} sends $(\mathbf{send}, id_m, id_{SEN}, id_{REC}, l_h(k) + l_m(k))$ and (2) when SEN sends $(id_m, [c])$ to REC in the simulation of the hybrid execution.

testpro ($j, \circ$): 1. It takes a tuple $(j, \circ) \in \{-1, \dots, P \} \times \{+, -, \times, /\}$ 2. Pick $x_r \xleftarrow{U} \mathbb{Z}_N^*$ and 3. If $l_m(k)$ is known, then pick $R_{l_h(k)+r} \xleftarrow{U} \mathbb{Z}_N^*, 1 \leq r \leq l_m(k)$ 4. If $\circ = \{ /\ }$ and $P_j(x_r, R_{l_h(k)+1}, \dots, R_{l_h(k)+l_m(k)}) \notin \mathbb{Z}_N^*$, Then return false Else return true	equalpro (j, i): 1. It takes a tuple $(j, i) \in \{-1, \dots, P \} \times \{-1, \dots, P \}$ 2. Pick $x_r \xleftarrow{U} \mathbb{Z}_N^*$ and 3. If $l_m(k)$ is known, then pick $R_{l_h(k)+r} \xleftarrow{U} \mathbb{Z}_N^*, 1 \leq r \leq l_m(k)$ 4. If $P_j(x_r, R_{l_h(k)+1}, \dots, R_{l_h(k)+l_m(k)}) \equiv_N$ $P_i(x_r, R_{l_h(k)+1}, \dots, R_{l_h(k)+l_m(k)})$ Then return true Else return false
--	---

TABLE 4.3: Definition of the routines **testpro** and **equalpro** for the size of the heading $l_h(k)$ for protocol π and the simulated ciphertexts $R_{l_h(k)+r}$.

Send: (1) $\mathcal{S}$ receives $(\text{send}, id_m, id_{SEN}, id_{REC}, l_h(k) + l_m(k))$ from $\mathcal{F}_{nce}$.

- If $\widetilde{REC}$ is corrupted:
 - $\mathcal{S}$ allows $(\text{receive}, id_m, id_{SEN}, [m])$ to be sent to $\widetilde{REC}$.
 - Since $\widetilde{REC}$ is corrupted, $\mathcal{S}$ learns m . Then it computes

$$c = \text{enc}^R(\text{pk}, id_m || id_{SEN} || id_{REC} || m)$$

and delivers $(id_m, [c])$ to $\mathcal{A}$, where $[c]$ are the pointers to the ciphertexts c_i .

- If $\widetilde{REC}$ is honest:
 - $\mathcal{S}$ adds to the sequence P of its simulation of $\mathcal{R}_{PRO}$ with $P^{(m_i)}(x) = R_{l_h(k)+i}$ and $1 \leq i \leq l_m(k)$, where $R_{l_h(k)+i}$ are random variables sampled uniformly at random from $\mathbb{Z}_N^*$.
 - $\mathcal{S}$ computes $\text{enc}^G(\text{pk}, id_m || id_{SEN} || id_{REC}) = (c_1, \dots, c_{l_h(k)})$.
 - $\mathcal{S}$ gives $\mathcal{A}$ the access to the pointers of the simulated ciphertext, i.e., $[c] = [c_1], \dots, [c_{l_h(k)}], [P^{(m_1)}(x)], \dots, [P^{(m_{l_m(k)})}(x)]$ through its simulation of $\mathcal{R}_{PRO}$.

If at a later moment REC is corrupted:

- If REC was corrupted
 - * $\mathcal{S}$ corrupts $\widetilde{REC}$ and learns $m = (m_1, \dots, m_{l_m(k)})$.

- * **S** commits to the correct ciphertext by computing

$$R_{l_h(x)+i} \leftarrow r_{l_h(k)+i}^2 \cdot x^{m_i},$$

for $r_{l_h(k)+i} \xleftarrow{U} \mathbb{Z}_N^*$ and $1 \leq i \leq l_m(k)$.

- * **S** aborts if the simulation was not perfect. Let this event be denoted as **abort**.
- * If **S** did not abort, then it sends a give up request on N to its challenger in the factoring game, and receives p and q .
- * **S** uses its simulation of $\mathbf{R}_{PRO}$ and $P_0(x) = 1$ to compile p and q as SLP's. In other words, **S** defines $P^{(p)}(x) := p$ and $P^{(q)}(x) := q$ and adds both to the sequence P .
- * **S** gives **A** the access to the pointers to the SLP-steps associated with p and q , i.e., $[P^{(p)}(x)]$ and $[P^{(q)}(x)]$, as the secret key **sk**.

If **S** does not abort, then it continues simulating $\mathbf{R}_{PRO}$, however with the fixed values $P^{(m_i)}(x) = r_{l_h(k)+i}^2 \cdot x^{m_i}$.

Receive: (2) **S** must make $\mathbf{F}_{nce}$ output $(\mathbf{receive}, id_m, id_{SEN}, [m])$ to $\widetilde{REC}$, when SEN sends $(id_m, [c])$ to REC .

- If SEN is honest

Then $(id_m, [c])$ was sent by **S** in response to what happen in the ideal world. Then, **S** must have received $(\mathbf{send}, id_m, id_{SEN}, id_{REC}, l_h(k) + l_m(k))$ from $\mathbf{F}_{nce}$. Therefore, **S** allows $(\mathbf{receive}, id_m, id_{SEN}, [m])$ to be sent to $\widetilde{REC}$ in the ideal execution.

Since we are not considering corruption of the receiver, this construction is enough.

If the simulation does not abort, then it is indistinguishable from the real case. Therefore, let us assume for the sake of contradiction that the simulation aborts. The event **abort** occurs when the simulated oracle $\mathbf{R}_{PRO}$ does not behave like **R**. That is, whenever either one of the two following events happens:

1. $\mathbf{abort}_{test}$: when **testpro** does not behave as **test**, or
2. $\mathbf{abort}_{equal}$: when **equalpro** does not behave as **equal**.

This translates in

$$\Pr[\mathbf{abort}] \leq \Pr[\mathbf{abort}_{test}] + \Pr[\mathbf{abort}_{equal}], \quad (4.1)$$

where $\mathbf{abort}_{test}$ and $\mathbf{abort}_{equal}$ are the events 1 and 2 respectively.

We estimate the probabilities $\Pr[\mathbf{abort}_{test}]$ and $\Pr[\mathbf{abort}_{equal}]$ separately¹. From this point, for convenience of the reader, assume that X are the values sampled by R_{PRO} on every operation. That is $X = (x_r, R_{l_h(k)+1}, \dots, R_{l_h(k)+l_m(k)})$.

Using the Lemma 4.4, a fixed SLP P_i and the committed values

$$X_c = (x, r_{l_h(k)+1}^2 \cdot x^{m_1}, \dots, r_{l_h(k)+l_m(k)}^2 \cdot x^{m_{l_m(k)}})$$

we have

$$\begin{aligned} \Pr[\mathbf{abort}_{test}] &\leq \Pr[(P_i(X_r) \notin \mathbb{Z}_N^* \text{ and } P_i(X_c) \in \mathbb{Z}_N^*) \text{ or} \\ &\quad (P_i(X_c) \notin \mathbb{Z}_N^* \text{ and } P_i(X_r) \in \mathbb{Z}_N^*) | X_1, \dots, X_{q(k)} \stackrel{U}{\leftarrow} (\mathbb{Z}_N^*)^{l_m(k)+1}] \\ &\leq 2q(k) \cdot \Pr[P_i(X) \notin \mathbb{Z}_N^* \text{ and } P_i(X') \in \mathbb{Z}_N^* | X, X' \stackrel{U}{\leftarrow} (\mathbb{Z}_N^*)^{l_m(k)+1}]. \end{aligned}$$

For, $1 \leq i \leq q(x)$, that is, for any $P_i \subseteq P$, we have

$$\begin{aligned} \Pr[\mathbf{abort}_{test}] &\leq 2q(k) \\ &\quad \cdot \sum_{i=0}^{q(k)} \Pr[P_i(X) \notin \mathbb{Z}_N^* \text{ and } P_i(X') \in \mathbb{Z}_N^* | X, X' \stackrel{U}{\leftarrow} (\mathbb{Z}_N^*)^{l_m(k)+1}] \\ &\leq 2q(k) \cdot (q(k) + 1) \cdot \max_{0 \leq i \leq q(k)} \{ \Pr[P_i(X) \notin \mathbb{Z}_N^* \text{ and} \\ &\quad P_i(X') \in \mathbb{Z}_N^* | X, X' \stackrel{U}{\leftarrow} (\mathbb{Z}_N^*)^{l_m(k)+1} \}. \end{aligned} \quad (4.2)$$

Similarly with probability $\Pr[\mathbf{abort}_{equal}]$, we have

$$\begin{aligned} \Pr[\mathbf{abort}_{equal}] &\leq 2q(k) \cdot (q(k)^2 + 3q(k) + 1) \\ &\quad \cdot \max_{-1 \leq i \leq j \leq q(k)} \{ \Pr[P_i(X) \equiv_N P_j(X) \text{ and} \\ &\quad P_i(X') \not\equiv_N P_j(X') | X, X' \stackrel{U}{\leftarrow} (\mathbb{Z}_N^*)^{l_m(k)+1}] \} \\ &\quad + \Pr[\mathbf{abort}_{test}]. \end{aligned} \quad (4.3)$$

¹These probabilities for single values x are estimated in detail in [22].

Equation 4.3 relates to the similar test from Lemma 4.5, that is a test which can be made sampling randomly from $\mathbb{Z}_N^*$. Similarly, Equation 4.2 relates to Lemma 4.6. Both tests are used in the factoring algorithm to be presented next.

We now construct the factoring algorithm. Let FACTOR be the factoring algorithm. The algorithm evaluates every SLP in the sequence P kept by R_{PRO} . FACTOR will run both tests from Lemmas 4.5 and 4.6, that is

1. Whenever R_{PRO} receives the r -th query operation $(i, j, \circ)$ where the ring operation $\circ$ is picked from the set $\{+, -, \times, /\}$, FACTOR samples $Y = (y_1, \dots, y_{l_m(k)+1}) \xleftarrow{U} (\mathbb{Z}_N^*)^{l_m(k)+1}$ and, for $0 \leq n \leq r$, computes $\gcd(P_n(Y), N)$ (Lemma 4.5).
2. Whenever R_{PRO} receives the r -th equality test $(i, j, \circ)$ with $\circ \in \{=\}$, FACTOR samples $Y = (y_1, \dots, y_{l_m(k)+1}) \xleftarrow{U} (\mathbb{Z}_N^*)^{l_m(k)+1}$ and computes $\gcd(P_i(Y) - P_j(Y), N)$ for $-1 \leq i < j \leq r$ (Lemma 4.6).

Assume that γ is the maximum value between the following two probabilities from Equations 4.3 and 4.2:

$$\max_{0 \leq i \leq q(k)} \{\Pr[P_i(X) \notin \mathbb{Z}_N^* \text{ and } P_i(X') \in \mathbb{Z}_N^* \mid X, X' \xleftarrow{U} (\mathbb{Z}_N^*)^{l_m(k)+1}]\}$$

and

$$\max_{-1 \leq i \leq j \leq q(k)} \{\Pr[P_i(X) \equiv_N P_j(X) \text{ and } P_i(X') \not\equiv_N P_j(X') \mid X, X' \xleftarrow{U} (\mathbb{Z}_N^*)^{l_m(k)+1}]\}.$$

Then, the algorithm FACTOR, and the Lemmas 4.5 and 4.6 give $\gamma \leq \varepsilon_{\text{FACT}}$. The maximum value γ upper bounds the failure probability

$$\begin{aligned} \Pr[\text{abort}] &\leq 4q(k) \cdot (q(k) + 1) \cdot \gamma + 2q(k) \cdot (q(k)^2 + 3q(k) + 1) \cdot \gamma \\ &\leq 2q(k) \cdot (q(k)^2 + 5q(k) + 3) \cdot \gamma, \end{aligned}$$

and Equation 4.1 upper bounds the probability of the event **abort**

$$\Pr[\text{abort}] \leq 2q(k) \cdot (q(k)^2 + 5q(k) + 3) \cdot \varepsilon_{\text{FACT}},$$

and this gives the end of the proof. $\square$

4.5 Summary

Likewise the earlier chapter, here we have introduced two novel models: the programmable and non-programmable GRM.

We have studied the use of programmability in GRM by proving that the Goldwasser-Micali encryption scheme is NINCE regarding only the corruption of the receiver and exposure of the secret key. Note the difference with the programmable GGM where ElGamal encryption scheme was shown to be NINCE regarding corruption of both the sender and the receiver.

This observation does not exclude the possibility that full (i.e., regarding corruption of receiver and sender) NINCE scheme is possible. However, we emphasize that this observation combined with the result of Nielsen [1] means that the programmable GRM is a stronger model.

Chapter 5

Final Remarks

The distinction between programmable and non-programmable models formalized by Nielsen [1] for ROM was based on the simulation paradigm. His proposed models differ by the possibility of the simulator of resorting to programmability. Moreover, it was noted that these two models are indeed different. While non-committing encryptions exist in the programmable ROM, it is not possible non-programmable ROM.

Similarly, Fischlin et al. [16] showed that without the programmability, in the reduction-centric paradigm, the RSA-FDH signature scheme cannot be proven EUF-CMA secure. Moreover, as different variants of weakened models for ROM have been proposed, it become clear that some security schemes may not be considered secure in weaker models. Non-programmable models are weaker as both results show, in particular for the case of RSA-FDH.

In Chapter 2, we study the question of the RSA-FDH in the non-programmable ROM constructed from the Fischlin et al. model. We considered a more general assumption than the regular one-wayness of the underlying trapdoor permutation: the one-more one-wayness. That is, while the regular one-wayness the adversary receives a single value to invert, the one-more one-wayness assumption the adversary receives an arbitrary number of ℓ values to invert and has access to $\ell - 1$ queries to an inversion oracle at its choice. Note that the one-more one-wayness assumption boils down to the regular one-wayness assumption when $\ell = 1$.

We have showed that RSA-FDH is EUF-CMA secure in the weakened variant of ROM, the non-programmable ROM, assuming the one-more one-wayness property

of the trapdoor permutation RSA. This is a reasonable assumption which has been studied before [24]. This observation is, in fact, not a contradiction to the negative result of Fischlin et al., it is, actually, a confirmation, in the sense that ℓ must be greater than 1 in order to show security. In particular, in Section 2.7, we discuss that under natural conditions, and when $q_s + k \cdot \ln(2) < \ell$, the RSA-FDH is EUF-CMA secure in the non-programmable ROM.

A technicality of our proof is that it still relies on a property of ROM. A closer look at proofs in ROM, would show that similar property has been used before. Namely, the assumption that we can always assume that whenever an adversary computes a forgery, the hash value of the forgery must have been produced by the random oracle at some point of the security game, as in Section 2.5.

This property, although its extensive use in provable security, has not been studied in detail.

Although the programmability of ROM has been used at length throughout the literature of provable security, it is defined only for the ROM. This is surprising, since some researchers, as Brown [49][Section 2.5.1] and Smart [52][Section 1], compare generic model and ROM. More notable works are those of Koblitz and Menezes [2, 30] and Dent [27]. In particular, the latest even claim that the GGM and ROM have similar weaknesses [27].

In general, our findings show that the ROM and generic models, GGM and GRM, share more similarities than observed before. In detail, this work introduced the notion of *programmability* for GGM and GRM. In comparison to the previous formalizations of the models, this observation, in itself, is a novel feature. Furthermore, our study have illustrated that, similarly to what happens in ROM, this property is crucial to show that in these models NINCE schemes are possible.

We point out that in both cases it is possible to prove despite the fact that the secret-key is exposed when the receiver is corrupted¹, which has been avoided in other schemes [56, Section 1.2 and 1.3]. In the case of the non-programmable GGM and GRM, the result in [1] rules out the existence of NINCE schemes.

One should have in mind that our proofs are in generic model, i.e., they provide the same caveats as the regular proofs in the ROM [26, 28] and generic model.

¹Which is a crucial step in the Nielsen’s information-theoretic argument in the ROM without programmability.

In particular, those discussed in [22], regarding multiplicative groups of finite fields [23] and proofs which rely on programmability [16]. On the other hand, there are certain elliptic curve groups that seem to fit into the GGM, because not many properties are known out of the group operations. For those cases, our result in the programmable GGM shows that NINCE schemes are possible in such a model.

The work of Nielsen confirms the intuition that proofs which rely on the programmability of the model are indeed more powerful, in the sense that more schemes can be proved secure. Therefore, from a more practical viewpoint our approach of using the programmability of the generic operation oracle is interesting because it offers a new observation and an alternative to the reduction-centric two-step security proofs in the generic models.

Bibliography

- [1] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *In proceedings of CRYPTO '02, LNCS series*, pages 111–126. Springer-Verlag, 2002.
- [2] Neal Koblitz and Alfred J. Menezes. Another look at "provable security". *Journal of Cryptology*, 20(1):3–37, 2007. doi: 10.1007/s00145-005-0432-z.
- [3] 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. ISSN 0097-5397. doi: <http://dx.doi.org/10.1137/0217017>.
- [4] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures—how to sign with RSA and Rabin. In Ueli Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer Berlin / Heidelberg, 1996. URL http://dx.doi.org/10.1007/3-540-68339-9_34. 10.1007/3-540-68339-9_34.
- [5] Jean-Sébastien Coron. On the exact security of full domain hash. In Mihir Bellare, editor, *Advances in Cryptology — CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235. Springer Berlin / Heidelberg, 2000. URL http://dx.doi.org/10.1007/3-540-44598-6_14. 10.1007/3-540-44598-6_14.
- [6] Jean-Sébastien Coron. Optimal security proofs for PSS and other signature schemes. In Lars Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 272–287. Springer Berlin / Heidelberg, 2002. URL http://dx.doi.org/10.1007/3-540-46035-7_18. 10.1007/3-540-46035-7_18.
- [7] Akira Numayama, Toshiyuki Isshiki, and Keisuke Tanaka. Security of digital signature schemes in weakened random oracle models. In Ronald Cramer,

- editor, *Public Key Cryptography – PKC 2008*, volume 4939 of *Lecture Notes in Computer Science*, pages 268–287. Springer Berlin / Heidelberg, 2008. URL http://dx.doi.org/10.1007/978-3-540-78440-1_16. 10.1007/978-3-540-78440-1_16.
- [8] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer Berlin / Heidelberg, 1997. URL http://dx.doi.org/10.1007/3-540-69053-0_18. 10.1007/3-540-69053-0_18.
- [9] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55:165–172, 1994. ISSN 0001-4346. URL <http://dx.doi.org/10.1007/BF02113297>. 10.1007/BF02113297.
- [10] Divesh Aggarwal and Ueli Maurer. Breaking RSA generically is equivalent to factoring. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 36–53. Springer Berlin / Heidelberg, 2009. URL http://dx.doi.org/10.1007/978-3-642-01001-9_2. 10.1007/978-3-642-01001-9_2.
- [11] Ueli Maurer. Abstract models of computation in cryptography. In Nigel Smart, editor, *Cryptography and Coding*, volume 3796 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin / Heidelberg, 2005. URL http://dx.doi.org/10.1007/11586821_1. 10.1007/11586821_1.
- [12] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. <http://eprint.iacr.org/>.
- [13] Ran Canetti, Uri Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, STOC '96, pages 639–648, New York, NY, USA, 1996. ACM. ISBN 0-89791-785-5. doi: <http://doi.acm.org/10.1145/237814.238015>. URL <http://doi.acm.org/10.1145/237814.238015>.
- [14] Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. In David Wagner, editor, *Advances in Cryptology - CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 21–38.

- Springer Berlin / Heidelberg, 2008. URL http://dx.doi.org/10.1007/978-3-540-85174-5_2. 10.1007/978-3-540-85174-5_2.
- [15] Hoeteck Wee. Zero knowledge in the random oracle model, revisited. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 417–434. Springer Berlin / Heidelberg, 2009. URL http://dx.doi.org/10.1007/978-3-642-10366-7_25. 10.1007/978-3-642-10366-7_25.
- [16] Marc Fischlin, Anja Lehmann, Thomas Ristenpart, Thomas Shrimpton, Martijn Stam, and Stefano Tessaro. Random oracles with(out) programmability. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 303–320. Springer Berlin / Heidelberg, 2010. URL http://dx.doi.org/10.1007/978-3-642-17373-8_18. 10.1007/978-3-642-17373-8_18.
- [17] Yevgeniy Dodis, Roberto Oliveira, and Krzysztof Pietrzak. On the generic insecurity of the full domain hash. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 449–466. Springer Berlin / Heidelberg, 2005. URL http://dx.doi.org/10.1007/11535218_27. 10.1007/11535218_27.
- [18] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. pages 62–73. ACM Press, 1993.
- [19] The Internet Engineering Task Force (IETF). Public-key cryptography standards (pkcs) #1: RSA cryptography specifications version 2.1, 2003. URL <http://tools.ietf.org/html/rfc3447>.
- [20] Tibor Jager and Jörg Schwenk. On the equivalence of generic group models. In Joonsang Baek, Feng Bao, Kefei Chen, and Xuejia Lai, editors, *Provable Security*, volume 5324 of *Lecture Notes in Computer Science*, pages 200–209. Springer Berlin / Heidelberg, 2008. URL http://dx.doi.org/10.1007/978-3-540-88733-1_14. 10.1007/978-3-540-88733-1_14.
- [21] Dan Boneh and Ramarathnam Venkatesan. Breaking RSA may not be equivalent to factoring. In Kaisa Nyberg, editor, *Advances in Cryptology — EUROCRYPT’98*, volume 1403 of *Lecture Notes in Computer Science*, pages 59–71. Springer Berlin / Heidelberg, 1998. URL <http://dx.doi.org/10.1007/BFb0054117>. 10.1007/BFb0054117.

- [22] Tibor Jager and Jörg Schwenk. On the analysis of cryptographic assumptions in the generic ring model. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 399–416. Springer Berlin / Heidelberg, 2009. URL http://dx.doi.org/10.1007/978-3-642-10366-7_24. 10.1007/978-3-642-10366-7_24.
- [23] Tibor Jager and Andy Rupp. The semi-generic group model and applications to pairing-based cryptography. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 539–556. Springer Berlin / Heidelberg, 2010. URL http://dx.doi.org/10.1007/978-3-642-17373-8_31. 10.1007/978-3-642-17373-8_31.
- [24] Bellare, Namprempre, Pointcheval, and Semanko. The one-more-RSA-inversion problems and the security of Chaum’s blind signature scheme. *Journal of Cryptology*, 16:185–215, 2008. ISSN 0933-2790. URL <http://dx.doi.org/10.1007/s00145-002-0120-1>. 10.1007/s00145-002-0120-1.
- [25] Mario Larangeira and Keisuke Tanaka. Reduction-centric non-programmable security proof for the full domain hash in the random oracle model. In DongHoon Lee and Moti Yung, editors, *Information Security Applications*, volume 7690 of *Lecture Notes in Computer Science*, pages 124–143. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-35415-1. doi: 10.1007/978-3-642-35416-8_10. URL http://dx.doi.org/10.1007/978-3-642-35416-8_10.
- [26] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004. ISSN 0004-5411. doi: <http://doi.acm.org/10.1145/1008731.1008734>.
- [27] Alexander Dent. Adapting the weaknesses of the random oracle model to the generic group model. In Yuliang Zheng, editor, *Advances in Cryptology — ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 95–104. Springer Berlin / Heidelberg, 2002. URL http://dx.doi.org/10.1007/3-540-36178-2_6. 10.1007/3-540-36178-2_6.
- [28] Ueli Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *Theory of Cryptography*, volume 2951

- of *Lecture Notes in Computer Science*, pages 21–39. Springer Berlin / Heidelberg, 2004. URL http://dx.doi.org/10.1007/978-3-540-24638-1_2. 10.1007/978-3-540-24638-1_2.
- [29] Marc Fischlin. A note on security proofs in the generic model. In Tatsuaki Okamoto, editor, *Advances in Cryptology — ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 458–469. Springer Berlin / Heidelberg, 2000. URL http://dx.doi.org/10.1007/3-540-44448-3_35. 10.1007/3-540-44448-3_35.
- [30] Neal Koblitz and Alfred Menezes. Another look at generic groups. Cryptology ePrint Archive, Report 2006/230, 2006. <http://eprint.iacr.org/>.
- [31] Mario Larangeira and Keisuke Tanaka. Programmability in the generic ring and group models. In *Journal of Internet Services and Information Security*, volume 1, pages 57–73. 2011.
- [32] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27:701–717, October 1980. ISSN 0004-5411. doi: <http://doi.acm.org/10.1145/322217.322225>. URL <http://doi.acm.org/10.1145/322217.322225>.
- [33] Gaëtan Leurent and Phong Nguyen. How risky is the random-oracle model? In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 445–464. Springer Berlin / Heidelberg, 2009. URL http://dx.doi.org/10.1007/978-3-642-03356-8_26. 10.1007/978-3-642-03356-8_26.
- [34] Akinori Kawachi, Akira Numayama, Keisuke Tanaka, and Keita Xagawa. Security of encryption schemes in weakened random oracle models. In Phong Nguyen and David Pointcheval, editors, *Public Key Cryptography – PKC 2010*, volume 6056 of *Lecture Notes in Computer Science*, pages 403–419. Springer Berlin / Heidelberg, 2010. URL http://dx.doi.org/10.1007/978-3-642-13013-7_24. 10.1007/978-3-642-13013-7_24.
- [35] Kazuki Yoneyama, Satoshi Miyagawa, and Kazuo Ohta. Leaky random oracle (extended abstract). In Joonsang Baek, Feng Bao, Kefei Chen, and Xuejia Lai, editors, *Provable Security*, volume 5324 of *Lecture Notes in Computer Science*, pages 226–240. Springer Berlin / Heidelberg, 2008. URL http://dx.doi.org/10.1007/978-3-642-03356-8_26.

- [//dx.doi.org/10.1007/978-3-540-88733-1_16](http://dx.doi.org/10.1007/978-3-540-88733-1_16). 10.1007/978-3-540-88733-1_16.
- [36] Dominique Unruh. Random oracles and auxiliary input. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 205–223. Springer Berlin / Heidelberg, 2007. URL http://dx.doi.org/10.1007/978-3-540-74143-5_12. 10.1007/978-3-540-74143-5_12.
- [37] Jesper Buus Nielsen, Jesper Buus Nielsen, and Jesper Buus Nielsen. Non-committing encryption is too easy in the random oracle model. www.brics.dk/RS/01/47/, 2001.
- [38] Oded Goldreich. *Foundations of Cryptography Volume I - Basic Tools*. Cambridge University Press, 2006.
- [39] Pascal Paillier. Impossibility proofs for RSA signatures in the standard model. In Masayuki Abe, editor, *Topics in Cryptology - CT-RSA 2007*, volume 4377 of *Lecture Notes in Computer Science*, pages 31–48. Springer Berlin / Heidelberg, 2006. URL http://dx.doi.org/10.1007/11967668_3. 10.1007/11967668_3.
- [40] Ivan Damgård and Maciej Koprowski. Generic lower bounds for root extraction and signature schemes in general groups. In LarsR. Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 256–271. Springer Berlin Heidelberg, 2002. ISBN 978-3-540-43553-2. doi: 10.1007/3-540-46035-7_17. URL http://dx.doi.org/10.1007/3-540-46035-7_17.
- [41] Gregor Leander and Andy Rupp. On the equivalence of rsa and factoring regarding generic ring algorithms. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology – ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 241–251. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-49475-1. doi: 10.1007/11935230_16. URL http://dx.doi.org/10.1007/11935230_16.
- [42] Ueli Maurer and Stefan Wolf. Lower bounds on generic algorithms in groups. In Kaisa Nyberg, editor, *Advances in Cryptology — EUROCRYPT’98*, volume 1403 of *Lecture Notes in Computer Science*, pages 72–84. Springer Berlin

- Heidelberg, 1998. ISBN 978-3-540-64518-4. doi: 10.1007/BFb0054118. URL <http://dx.doi.org/10.1007/BFb0054118>.
- [43] Ueli M. Maurer and Stefan Wolf. The relationship between breaking the diffie–hellman protocol and computing discrete logarithms. *SIAM J. Comput.*, 28(5):1689–1721, May 1999. ISSN 0097-5397. doi: 10.1137/S0097539796302749. URL <http://dx.doi.org/10.1137/S0097539796302749>.
- [44] Ueli M. Maurer. Towards the equivalence of breaking the diffie-hellman protocol and computing discrete algorithms. In *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, volume 839 of *Lecture Notes in Computer Science*, pages 271–281. Springer, 1994. doi: 10.1007/3-540-48658-5_26.
- [45] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.
- [46] Bert Boer. Diffie-hellman is as strong as discrete log for certain primes. In Shafi Goldwasser, editor, *Advances in Cryptology — CRYPTO' 88*, volume 403 of *Lecture Notes in Computer Science*, pages 530–539. Springer New York, 1990. ISBN 978-0-387-97196-4. doi: 10.1007/0-387-34799-2_38. URL http://dx.doi.org/10.1007/0-387-34799-2_38.
- [47] Dan Boneh and Richard Lipton. Algorithms for black-box fields and their application to cryptography. In Neal Koblitz, editor, *Advances in Cryptology — CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 283–297. Springer Berlin / Heidelberg, 1996. URL http://dx.doi.org/10.1007/3-540-68697-5_22. 10.1007/3-540-68697-5_22.
- [48] Alexander W. Dent. The hardness of the dhk problem in the generic group model. Cryptology ePrint Archive, Report 2006/156, 2006. <http://eprint.iacr.org/>.
- [49] Daniel R.L. Brown. Generic groups, collision resistance, and ecdsa. *Designs, Codes and Cryptography*, 35:119–152, 2005. ISSN 0925-1022. doi: 10.1007/s10623-003-6154-z. URL <http://dx.doi.org/10.1007/s10623-003-6154-z>.
- [50] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the sdh assumption in bilinear groups. *Journal of Cryptology*, 21:149–177,

2008. ISSN 0933-2790. doi: 10.1007/s00145-007-9005-7. URL <http://dx.doi.org/10.1007/s00145-007-9005-7>.
- [51] ClausPeter Schnorr and Markus Jakobsson. Security of signed elgamal encryption. In Tatsuaki Okamoto, editor, *Advances in Cryptology — ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 73–89. Springer Berlin Heidelberg, 2000. ISBN 978-3-540-41404-9. doi: 10.1007/3-540-44448-3_7. URL http://dx.doi.org/10.1007/3-540-44448-3_7.
- [52] N.P. Smart. The exact security of ecies in the generic group model. In Bahram Honary, editor, *Cryptography and Coding*, volume 2260 of *Lecture Notes in Computer Science*, pages 73–84. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-43026-1. doi: 10.1007/3-540-45325-3_8. URL http://dx.doi.org/10.1007/3-540-45325-3_8.
- [53] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle diffie-hellman assumptions and an analysis of dhies. In David Naccache, editor, *Topics in Cryptology — CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 143–158. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-41898-6. doi: 10.1007/3-540-45353-9_12. URL http://dx.doi.org/10.1007/3-540-45353-9_12.
- [54] Daniel R. L. Brown. Breaking rsa may be as difficult as factoring. Cryptology ePrint Archive, Report 2005/380, 2005. <http://eprint.iacr.org/>.
- [55] Antoine Joux, David Naccache, and Emmanuel Thomé. When e-th roots become easier than factoring. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 13–28. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-76899-9. doi: 10.1007/978-3-540-76900-2_2. URL http://dx.doi.org/10.1007/978-3-540-76900-2_2.
- [56] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 1–35. Springer Berlin / Heidelberg, 2009. URL http://dx.doi.org/10.1007/978-3-642-01001-9_1. 10.1007/978-3-642-01001-9_1.