Non-Interactive Key Exchange and Key Assignment Schemes

Eduarda Simões Veloso Freire

Thesis submitted to the University of London
for the degree of Doctor of Philosophy

Information Security Group
School of Information Security and Mathematics
Royal Holloway, University of London

2014

Declaration

These doctoral studies were conducted under the supervision of Professor Kenneth G. Paterson.

The work presented in this thesis is the result of original research carried out by myself, in collaboration with others, whilst enrolled in the Department of Mathematics as a candidate for the degree of Doctor of Philosophy. This work has not been submitted for any other degree or award in any other university or educational establishment.

Eduarda Simões Veloso Freire
January, 2014

Abstract

This thesis is divided into two distinct parts. The first part of this thesis studies non-interactive key exchange schemes in two different settings: the public key setting and the identity-based setting. Loosely speaking, a non-interactive key exchange (NIKE) scheme allows two users to compute a unique shared key without any interaction. Our work is motivated by the scant attention that this primitive has received since the major contribution in the ground-breaking paper of Diffie and Hellman.

In the public key setting, we assume that any user can compute a public/private key pair and the public keys are registered with a Certification Authority (CA). A user A can compute a shared key with user B by using its own private key sk_A and B's public key pk_B, along with some public parameters. We provide different security models for NIKE and explore the relationships between them. Our models consider the challenging setting where an adversary can introduce arbitrary public keys in the system. We give constructions for secure NIKE, with respect to those security models, in the random oracle model based on the hardness of factoring, and in the standard model based on the hardness of a variant of the Decisional Bilinear Diffie-Hellman problem for asymmetric pairings. We also study the relationship between NIKE and public key encryption (PKE), showing that a secure NIKE can be generically converted into an IND-CCA secure PKE scheme.

In the identity-based setting, there is a Trusted Authority (TA) who holds a master secret key and a master public key. The public key of a user is some unique information that identifies a user, called the identity. The private key for each user is computed by the TA, who uses its master secret key and master public key together with the user identity to derive the user's private key. Using multilinear maps, we obtain the first identity-based non-interactive key exchange scheme (ID-NIKE) secure in the standard model. The scheme is a standard-model version of the Sakai-Ohgishi-Kasahara ID-NIKE scheme. In addition, we derive a fully-secure hierarchical version of our ID-NIKE scheme. Our hierarchical ID-NIKE scheme is the first such scheme with full security in either the random oracle model or the standard model.

The second part of this thesis is concerned with the construction of hierarchical key assignment schemes. Such schemes can be used to enforce access control policies by cryptographic means. We present new, enhanced security models for hierarchical key assignment schemes and give simple, efficient and strongly key indistinguishable

secure constructions that can be used for arbitrary hierarchies. Our constructions use pseudorandom functions and forward-secure pseudorandom generators as building blocks. We compare instantiations of our constructions with state-of-the-art hierarchical key assignment schemes, demonstrating that our new schemes possess an attractive trade-off between storage requirements and efficiency of key derivation.

Contents

List of Figures

Abbreviations

AHF:	Admissible Hash Function
BBS:	Blum-Blum-Shub
BDH:	Bilinear Diffie-Hellman
CA:	Certification Authority
CCA:	Chosen Ciphertext Attack
CDH:	Computational Diffie-Hellman
CRT:	Chinese Remainder Theorem
DBDH:	Decisional Bilinear Diffie-Hellman
DBDH-2:	Decisional Bilinear Diffie-Hellman for Type 2 Pairings
DDH:	Decisional Diffie-Hellman
DEM:	Data Encapsulation Mechanism
DL:	Discrete Logarithm
DSDH:	Double-Strong Diffie-Hellman
FS-PRG:	Forward-Secure Pseudorandom Generator
GGH:	Garg-Gentry-Halevi
H-ID-NIKE:	Hierarchical Identity-based Non-Interactive Key Exchange
HKAS:	Hierarchical Key Assignment Scheme
IBE:	Identity-based Encryption
ID-NIKE:	Identity-based Non-Interactive Key Exchange
IND-CCA:	Indistinguishability against Chosen-Ciphertext Attacks
IND-CPA:	Indistinguishability against Chosen-Plaintext Attacks
KEM:	Key Encapsulation Mechanism
MAC:	Message Authentication Code
MDDH:	Multilinear Decisional Diffie-Hellman
MPHF:	Multilinear Programmable Hash Function
NIKE:	Non-Interactive Key Exchange

OTS:	One-Time Signature
PHF:	Programmable Hash Function
PKE:	Public Key Encryption
PKI:	Public Key Infrastructure
PPT:	Probabilistic Polynomial-Time
PRF:	Pseudorandom Function
PRG:	Pseudorandom Generator
QR:	Quadratic Residuosity
ROM:	Random Oracle Model
SDH:	Strong Diffie-Hellman
S-NIKE:	Simplified NIKE
SOK:	Sakai-Ohgishi-Kasahara
TA:	Trusted Authority

Notation

We include here a list of symbols and notations that might be useful for reference. All other symbols and notations will be introduced at their first use.

$\mathbb{Z}, \mathbb{N}, \mathbb{R}$	sets of integers, non-negative integers and real numbers
$k \in \mathbb{N}$	security parameter (also denoted by 1^k, the string of k ones)
$\mathbb{Z}_n$	set of all integers modulo n
$\mathbb{Z}_n^*$	subset of $\mathbb{Z}_n$ formed by all elements which are relatively prime to n
$\{0,1\}^n$	set of all binary strings of length n
$\{0,1\}^*$	set of all binary strings of unspecified length
$x\|\|y$	concatenation of strings x and y
$x \oplus y$	exlusive-or (XOR) of strings x and y
$a \mid b$	integer a divides integer b
$\|X\|$	number of elements in a set X, or order of group if X is a group
$\|\|x\|\|$	length of bit string x
$\|a\|$	absolute value of an element a
$\text{ord}(a)$	multiplicative order of element a in a specified group
$\phi : \mathbb{N} \to \mathbb{N}$	Euler's totient function
$[n]$	set of integers $1, \ldots, n$
$\lfloor x \rfloor$	largest integer not greater than x
$\{n_i\}_{i \in [q]}$	the set $\{n_1, \ldots, n_q\}$
$s \leftarrow S$	process of selecting s uniformly at random from a set/group S
$y \leftarrow \mathcal{A}(x)$	process of running an algorithm $\mathcal{A}$ on input x and assigning y the result
$x \leftarrow y$	process of assigning the value y to a variable x
$\mathcal{A}^{\mathsf{F}}$	an adversary $\mathcal{A}$ with oracle access to F
$\perp$	error/rejection symbol
$\neg, \wedge, \vee$	negation, logical conjunction, logical disjunction
$:=$	assignment symbol

Publications

This thesis is primarily based on four papers that have been published in international peer-reviewed conference proceedings. Their full citations are provided below.

1. Eduarda S. V. Freire, Dennis Hofheinz, Kenneth G. Paterson, and Christoph Striecks. Programmable hash functions in the multilinear setting. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 513–530. Springer, 2013

2. Eduarda S. V. Freire, Kenneth G. Paterson, and Bertram Poettering. Simple, efficient and strongly KI-secure hierarchical key assignment schemes. In Ed Dawson, editor, *CT-RSA*, volume 7779 of *Lecture Notes in Computer Science*, pages 101–114. Springer, 2013

3. Eduarda S. V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. Non-interactive key exchange. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 254–271. Springer, 2013

4. Eduarda S. V. Freire and Kenneth G. Paterson. Provably secure key assignment schemes from factoring. In Udaya Parampalli and Philip Hawkes, editors, *ACISP*, volume 6812 of *Lecture Notes in Computer Science*, pages 292–309. Springer, 2011

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my Ph.D. supervisor Kenny Paterson for his invaluable guidance and support throughout these past four years. It has been a privilege doing research under his supervison and without him this thesis would not have been possible. He proposed research projects and gave continuous feedback on my work, guiding me to achieve the highest standards in research. He helped me with presenting results in the best way possible, and he introduced me to numerous researchers in cryptography, enabling me to start work in collaboration with some of them. He has always been there whenever I have needed help; more recently, assisting me with career related advice. Thank you so much Kenny for having given me the opportunity to do a Ph.D. under your supervision and for being such an outstanding supervisor and co-author!

Many thanks to the Information Security Group Ph.D. research programme and to the director of the ISG, Keith Martin, for all the opportunities and support provided during my studies. I am grateful to my previous and current advisors Chris Mitchell and Jason Crampton who accompanied my development via the annual reviews. Thanks to all my friends and colleagues for their support and for promoting such an enjoyable environment in the department. I particularly remember how dedicated was Saif Al-Kuwari to supporting me in the preparation of my first annual review meeting.

I also would like to thank Eike Kiltz, Dennis Hofheinz, Julia Hesse, Christoph Striecks and Bertram Poettering for being amazing co-authors of my publications. It has been a real pleasure working with all of you. Vielen Dank! My special thanks to Dennis for having invited me to KIT twice, giving me the opportunity to learn so much with him and to meet all the nice people from Karlsruhe.

I would like to extend my gratitude to all other people from the cryptographic community who contributed with my development as a researcher, and also to my external and internal examiners Nigel Smart and Keith Martin for providing such a pleasant viva experience.

My sincere thanks also go to my friend and previous landlady Diana Allen, who warmly welcomed me to her house in the beginning of my Ph.D. and since then has always been so kind and supportive in many ways.

Back in Brazil, I would like to thank my Masters supervisor Ricardo Menezes Campello de Souza and my previous professor Valdemar Cardoso da Rocha Junior for encouraging me to pursue my Ph.D. studies at RHUL, for directly helping me in the process of receiving a scholarship, and for always being supportive through

these years. I also thank Cecílio José Lins Pimentel for helping with the scholarship application, and Márcia Mahon Campello de Souza for always being encouraging.

I thank CAPES for providing the financial support for my Ph.D. studies.

My heartfelt thanks to all my family members, specially my mother Mariza, my late father Eduardo, my sisters Camila and Milena, uncle Gabriel, and last but not least, my exceptional boyfriend Robert, for their unconditional love and support. I simply cannot thank them enough.

Chapter 1

Introduction

Contents

In this chapter, we give some motivation for our research and present the overall structure of this thesis with a summary of our contributions.

1.1 Motivation

This thesis addresses two distinct topics in cryptography: non-interactive key exchange and hierarchical key assignment schemes.

Part I of this thesis deals with the study of non-interactive key exchange schemes in two different settings: the public key setting and the identity-based setting.

In 1976, Diffie and Hellman in their ground-breaking paper "New directions in cryptography" [53], proposed a "public key distribution system". This system, famously known as the *Diffie-Hellman (DH) protocol*, is frequently seen as an *interactive* key exchange scheme; it allows two users to establish a shared key via communication over an insecure channel. Loosely, in this view, the DH protocol works as follows. We assume the existence of a group $\mathbb{G}$ of prime order p and with generator g. In order to agree on a shared key, user Alice randomly chooses an integer a from $\mathbb{Z}_p$ as her private key and user Bob similarly chooses a random integer b from $\mathbb{Z}_p$. Now, Alice computes her public key $g^a \pmod p$, which she sends to Bob, and Bob computes $g^b \pmod p$, which he sends to Alice. After receiving each other's public keys, Alice and Bob compute their shared key $K_{AB} = g^{ab} \pmod p$, which Alice computes as $K_{AB} = (g^b)^a \pmod p$ and Bob computes as $K_{AB} = (g^a)^b \pmod p$. The idea

behind this protocol is that, unless one can efficiently compute discrete logarithms in $\mathbb{G}$, it should be computationally infeasible for a potential adversary eavesdropping the communication between Alice and Bob to compute their shared key K_{AB}. However, it is well known that the DH protocol has a major shortcoming. Namely, it is vulnerable to a man-in-the-middle attack; an eavesdropper intercepts Alice's public key and sends its own public key to Bob. When Bob sends his public key, the eavesdropper now intercepts that communication and substitutes Bob's public key with its own public key. Alice and Bob, instead of agreeing on a shared key between themselves, will thus unknowingly each agree on a shared key with the eavesdropper.

Although frequently seen as an interactive key exchange scheme, the DH protocol was in fact originally proposed as a, what we call, *non-interactive* key exchange scheme (NIKE) in the public key setting; users (instead of exchanging public keys) store their public keys (along with their respective identities) in a public file, or in more practical terms, they register their public keys with a Certification Authority (CA). In order to compute a shared key with Bob, Alice first retrieves Bob's public key from the public file, while Bob retrieves Alice's public key. Alice and Bob compute a shared key as before, but no interaction between them is required. The DH protocol seen in this way suffers from a similar security problem as before, if we allow an adversary to register arbitrary public keys against users of its choice. So for example, the adversary "steals" Bob's public key and registers it as the public key of a third, corrupt, user. Now, we see that the shared key between Alice and the corrupt user, K_{AC}, is identical to K_{AB}. Thus, the adversary can obtain the shared key between two honest users, Alice and Bob, by simply stealing $K_{AC} = K_{AB}$. This security issue can be avoided if Alice and Bob hash that key along with their identities. In this case we say the scheme is secure in the random oracle model (see Section 2.2.3 for the random oracle model).

Interestingly, since its appearance in the Diffie-Hellman paper [53], the NIKE primitive has not received much attention. In the *public key setting*, to the best of our knowledge, the first work that provided a formal security model for non-interactive key exchange was a paper due to Cash *et al.*, in 2008 [40]. Then the questions that arise are: How realistic is this security model? What other options for security models can we have? Does there exist any NIKE scheme which is secure in the standard model with respect to a security model that allows an adversary to arbitrarily register public keys against users of its choice? (One has to consider that in practice

it is common that CAs do not operate so rigorously and might, for example: not properly check the identity of a user who wants to register a public key; not require proof of possession of corresponding private keys; or even not check validity of public keys.) In this thesis, we systematically study NIKE, addressing all these questions.

On the other hand, as far as we are aware, the first and only efficient and secure non-interactive key exchange scheme in the *identity-based setting* (ID-NIKE), was due to Sakai, Ohgishi and Kasahara, in 2000 [122]. (In the identity-based setting, publicly known strings identifying users are used as their public keys, and users' private keys are computed and distributed to the corresponding users by a Trusted Authority (TA).) The SOK scheme has been proven secure in a security model by Dupont and Enge [57] and subsequently in a stronger security model by Paterson and Srinivasan [112]. However, security for the SOK scheme is only achieved in the random oracle model. In this respect, the goal of this thesis is to provide the *first* ID-NIKE scheme with security in the *standard model*. We go further and examine the ID-NIKE primitive in the *hierarchical setting*, i.e. where users are organized in a hierarchy and every user can derive private keys for all its descendants in the hierarchy; as before, any user in the system should be able to securely and non-interactively agree on a shared key with any other user in the system.

Part II of this thesis is devoted to the study of hierarchical key assignment schemes (HKASs). Such schemes are methods for implementing *access control policies* by assigning encryption keys and private information to each class in a hierarchy in such a way that the private information assigned to a class, along with some public information, can be used to derive encryption keys to all the descendant classes in the hierarchy. Our main concern is that previous HKASs either lack any formal security analysis or have security based on not so realistic security models. Hierarchical key assignment schemes were first formalized by Akl and Taylor in 1983 [5], but work to formally analyse the security of such schemes only started in 2005 with Atallah *et al.* [8], who proposed two different security models. We argue that their security models do not capture the broadest range of realistic attacks; they do not allow an adversary, whose goal is to either compute the encryption key of a target class of its choice or distinguish that key from random, to gain access to keys for ancestors of the target class. This is a weakness in their security models as these keys might leak through usage, and their compromise need not trivially enable the computation of key for the target class. Therefore, in this thesis we address the issue of constructing

provably secure schemes with respect to stronger security models, which provide that additional compromise capability to the adversary.

Another contribution that we make in Part II of this thesis is to construct secure hierarchical key assignment schemes for arbitrary hierarchies in a simple manner. Recently, Crampton *et al.* [48] proposed an intriguing approach to constructing HKASs for arbitrary hierarchies using *chain partitions.* In this approach an arbitrary hierarchy is partitioned into a collection of chains and the scheme for the arbitrary hierarchy is built by combining, in a particular way, the schemes for each of the chains. However, that construction came with no formal security analysis. In this thesis we analyse the chain partition construction of [48], showing that the security of the scheme for a single chain implies the security of the scheme for an arbitrary hierarchy, enabling us to focus on the construction of simple schemes for single chains. We show how to construct such single chain schemes using pseudorandom functions and forward-secure pseudorandom generators.

1.2 Thesis Structure and Summary of Contributions

The remainder of this thesis is organised as follows.

Chapter 2: This chapter provides some background material relevant for the understanding of the subsequent chapters of this thesis. In particular, it gives an overview of the mathematics and cryptographic primitives required in the constructions of our cryptographic schemes presented in Parts I and II of this thesis. As this thesis follows the provable security paradigm, we also provide in this chapter an introduction to the topic of provable security and its proof techniques.

The main contributions of this thesis are organized in two separate parts: Part I, which consists of Chapters 3 and 4, focuses on the study of the non-interactive key exchange primitive; Part II, which consists of Chapter 5, is concerned with the study of hierarchical key assignment schemes. We expand on the contributions presented on each of these chapters next.

Chapter 3: This chapter is on NIKE in the public key setting. We start our contributions in this topic by exploring different security models for NIKE and their

relationships. Our *default* security models minimize the assumptions about the Certification Authority (CA), allowing the adversary to register arbitrary public keys in the system. We prove that all these models are polynomially equivalent, enabling us to analyse NIKE schemes in the simplest of the security models. We then provide two provably secure constructions for NIKE schemes: the first is based on pairings and is provably secure in the standard model, while the second is an adaptation of the hashed DH protocol to the group of signed quadratic residues and has security in the random oracle model. We continue our contributions on NIKE, showing how to construct an IND-CCA secure public key encryption scheme from a secure NIKE scheme.

The work presented in this chapter appears in the following publication.

- Eduarda S. V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. Non-interactive key exchange. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 254–271. Springer, 2013

In this thesis the proof of equivalence between the NIKE security models is given in a single theorem (Theorem 3.1) as opposed to three separate theorems as in [63] (the publication above).

Chapter 4: This chapter presents the *first ever* provably secure ID-NIKE scheme with security in the standard model. Our scheme is based on multilinear maps; we use multilinear maps to construct a new variant of programmable hash functions (PHFs), which we call multilinear PHFs (MPHFs), and then use such an MPHF to construct our standard model secure ID-NIKE scheme. Using MPHFs once again, we construct the *first ever* hierarchical ID-NIKE (H-ID-NIKE) scheme to achieve full security either in the random oracle model or in the standard model.

Most of the results of this chapter were published and presented at *CRYPTO 2013*:

- Eduarda S. V. Freire, Dennis Hofheinz, Kenneth G. Paterson, and Christoph Striecks. Programmable hash functions in the multilinear setting. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 513–530. Springer, 2013

Section 4.4 contains a construction towards a standard model secure ID-NIKE scheme, which is more efficient than the one introduced in the above publication – it needs only pairings instead of multilinear maps – but only offers security in a model where the adversary is restricted to make a bounded number of queries. This part of Chapter 4 is joint work with Dennis Hofheinz.

We now describe our contributions on hierarchical key assignment schemes.

Chapter 5: We start off this chapter by proposing new and stronger security models for HKASs than the ones proposed by Atallah *et al.* [8]. We call them S-KR and S-KI, for strong key-recovery security and strong key-indistinguishability security, respectively. We give two simple and efficient constructions for HKAS which we prove secure in our S-KI security model: the first construction is based on pseudorandom functions (PRFs), while the second is based on forward-secure pseudorandom generators (FS-PRGs). Both of our constructions are designed for totally ordered hierarchies and can be combined with the chain partition construction of Crampton *et al.* [48], which we formally analyse in this chapter, to produce HKASs for arbitrary hierarchies. In this chapter, we also provide a comparison between our constructions and some provably secure HKASs from the literature.

Most of the work presented in this chapter appears in the following publication.

- Eduarda S. V. Freire, Kenneth G. Paterson, and Bertram Poettering. Simple, efficient and strongly KI-secure hierarchical key assignment schemes. In Ed Dawson, editor, *CT-RSA*, volume 7779 of *Lecture Notes in Computer Science*, pages 101–114. Springer, 2013

The latter publication is in turn an extension of the publication below.

- Eduarda S. V. Freire and Kenneth G. Paterson. Provably secure key assignment schemes from factoring. In Udaya Parampalli and Philip Hawkes, editors, *ACISP*, volume 6812 of *Lecture Notes in Computer Science*, pages 292–309. Springer, 2011

Chapter 6: This is the concluding chapter. We summarize the overall contributions of this thesis and discuss a number of open problems and possible extensions of our work.

Chapter 2

Preliminary Topics

Contents

In this chapter we give an overview of some relevant concepts and background material that help in understanding the remainder of this thesis, making it as self-contained as possible.

2.1 Mathematical Background

2.1.1 Computational Complexity Theory

Computational complexity theory is the field of study that classifies computational problems according to the resources required to solve them. The resources may include, for example, time, storage space and randomness, with time usually being the main focus. Typically, computational complexity theory examines how those resources scale with the size of the problem, say k. For example, the time required to solve a problem may increase with some polynomial function of k or it may scale with some exponential function of k. This section provides some basic terminology and definitions used in this thesis.

An *algorithm* is a computational procedure that takes a variable input and produces an output. A *deterministic* algorithm is an algorithm which, for a given input, follows the same execution path every time it is executed – hence, for a given input, the algorithm will always produce the same output (the output is predefined). An algorithm is said to be *probabilistic* or *randomized* if its execution path depends not only on its input, but also on some random bits – hence, for the same input, the output of a probabilistic algorithm may differ each time it is executed.

In this thesis we use the term *cryptographic primitives*, or simply primitives, to denote basic cryptographic algorithms used in the construction of a more complex set of algorithms, referred to as a *cryptographic scheme*, or simply a scheme. So the term cryptographic primitives will be used when we refer to the basic building blocks of a cryptographic scheme. Note that the distinction between those two terms is quite arbitrary; they are often interchangeable in the literature.

The *running time* of an algorithm on a particular input is the number of "steps" executed by the algorithm before it terminates. A step is usually interpreted as a bit operation, but it might also be convenient to see a step as a basic operation such as an addition, a multiplication, a comparison, a machine clock cycle, etc. The *worst-case running time* of an algorithm is an upper bound on its running time for any input. In contrast, the *expected running time* of an algorithm is the average running

time over all inputs of a fixed size. Both worst-case and expected running times are expressed as functions of the input size. However, these functions are often highly dependent on the low-level details of the basic operations and thus it is difficult to calculate the exact running time of an algorithm. To help us simplify the calculation, we often rely on approximations of the running time. A very useful and well known notation in this regard is the big-$\mathcal{O}$ notation, which is used to express an asymptotic upper bound (usually in terms of some known function, such as a polynomial, an exponential function, or a logarithmic function) for a certain function.

Definition 2.1 (Big-$\mathcal{O}$ notation)**.** *Let f and g be two functions from the positive integers to the real numbers, then $f(k) = \mathcal{O}(g(k))$ if there is a positive constant c and a positive integer k_0 such that $0 \leq f(k) \leq cg(k)$ for all $k \geq k_0$.*

Definition 2.2 (Polynomial-time algorithm)**.** *A* polynomial-time algorithm *is an algorithm whose worst-case running time function is of the form $\mathcal{O}(k^c)$, where k is the input size and c is a constant.*

Any algorithm whose running time is not of the form $\mathcal{O}(k^c)$ is called a *superpolynomial-time algorithm*. This includes *exponential-time algorithms*, which are algorithms whose worst-case runnning time functions are of the form $\mathcal{O}(2^{k^c})$.

Some other useful notations in the framework of complexity theory are the notions of a *negligible function*, a *noticeable function* and an *overwhelming function*.

Informally, a negligible function is a function that grows slower than any inverse polynomial.

Definition 2.3 (Negligible function)**.** *A function $\epsilon(k)$ is* negligible *in the parameter k if for every positive constant c, there exists an integer k_0 such that $\epsilon(k) \leq k^{-c}$ for all $k \geq k_0$.*

Definition 2.4 (Overwhelming function)**.** *A function $\tau(k)$ is* overwhelming *in the parameter k if $1 - \tau(k)$ is negligible.*

Definition 2.5 (Noticeable function)**.** *A function $\mu(k)$ is* noticeable *in the parameter k if for every positive constant c, there exists an integer k_0 such that $\mu(k) \geq k^{-c}$ for all $k \geq k_0$.*

In cryptography, it is a common approach to view the complexity of algorithms as well as their probability of success as functions of the size of the problem, called the *security parameter*. The security parameter, k, is usually expressed in unary representation 1^k (meaning a k-long string of 1's). Then the notion of *efficient* algorithms is equivalent to polynomial-time algorithms, while exponential-time algorithms are regarded as *inefficient* algorithms. A problem that cannot be solved in polynomial-time is called *intractable* or *infeasible*. Furthermore, the notion of small success probability of an algorithm is substituted by negligible probability of success.

Definition 2.6 (Statistically close distributions)**.** *Given two distributions D_1, D_2, the* statistical distance *between them is defined as*

$$\Delta(D_1, D_2) := \frac{1}{2} \sum_x |\Pr[D_1 = x] - \Pr[D_2 = x]|,$$

where $\Pr[D_i = x]$ means the probability that a draw from D_i lands in x (for $i \in \{1, 2\}$). We say that D_1, D_2 are statistically ϵ-close *if $\Delta(D_1, D_2) \leq \epsilon$. If ϵ is negligible, we simply say that D_1, D_2 are* statistically close*.*

2.1.2 Abstract Algebra and Some Special Groups

Throughout this thesis we make free use of basic concepts of abstract algebra. For completeness, we recapitulate here some of these concepts. Further information can be found in [107, 133].

We denote the set of natural numbers by $\mathbb{N}$, the set of integers by $\mathbb{Z}$ and the set of integers modulo n by $\mathbb{Z}_n$. The subset of $\mathbb{Z}_n$ formed by the elements of $\mathbb{Z}_n$ which are relatively prime to n is denoted by $\mathbb{Z}_n^*$, that is, $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \mid \gcd(a, n) = 1\}$. For an integer $n \geq 1$, the number of integers in the interval $\{1, \ldots, n\}$ that are relatively prime to n is denoted by $\phi(n)$, where the function ϕ is called the *Euler phi function*. We let $\mathbb{G}$ denote a group, i.e. $\mathbb{G}$ consists of a set with a binary operation satisfying the group axioms: closure, associativity, identity and invertibility. Henceforth we will adopt the multiplicative group notation for the group operation.

Definition 2.7 (Order of a group)**.** *The* order *of a group $\mathbb{G}$ is the number of elements in the group, denoted by $|\mathbb{G}|$. A group $\mathbb{G}$ is finite if $|\mathbb{G}|$ is finite.*

Definition 2.8 (Cyclic group)**.** *A group* $\mathbb{G}$ *is* cyclic *if there is an element* $g \in \mathbb{G}$ *such that for each* $a \in \mathbb{G}$ *there is an integer* i *with* $a = g^i$*. Such an element* g *is called a* generator *of* $\mathbb{G}$*. We write* $\mathbb{G} = \langle g \rangle = \{g^i \mid i \in \mathbb{Z}\}$*.*

We say that a non-empty subset $\mathbb{H}$ of a group $\mathbb{G}$ is a subgroup of $\mathbb{G}$ if $\mathbb{H}$ also forms a group under the operation of $\mathbb{G}$. If $\mathbb{G}$ is cyclic, so is any of its subgroups $\mathbb{H}$. Furthermore, for an element $a \in \mathbb{G}$ the set of all powers of a forms a cyclic subgroup of $\mathbb{G}$, denoted by $\langle a \rangle$. We say that the order of a, denoted $\text{ord}(a)$, is equal to the order of the cyclic subgroup generated by a, $|\langle a \rangle|$. Note then that the order of a generator g of $\mathbb{G}$ (if it exists) is equal to the order of the group, that is $\text{ord}(g) = |\langle g \rangle| = |\mathbb{G}|$.

Theorem 2.1 (Lagrange)**.** *For any finite group* $\mathbb{G}$*, the order of every subgroup* $\mathbb{H}$ *of* $\mathbb{G}$ *divides the order of* $\mathbb{G}$*. That is,* $|\mathbb{H}|$ *divides* $|\mathbb{G}|$*. Hence, for any element* $a \in \mathbb{G}$*,* $|\langle a \rangle|$ *divides* $|\mathbb{G}|$*.*

Theorem 2.2. *Let* $\mathbb{G}$ *be a group. If* $|\mathbb{G}|$ *is prime, then* $\mathbb{G}$ *is cyclic. Moreover, every element* $g \in \mathbb{G}$*, other than the identity, is a generator of* $\mathbb{G}$*.*

Theorem 2.2 is a direct consequence of Lagrange's theorem.

Definition 2.9 (Abelian group)**.** *An* abelian group*, also called a* commutative group*, is a group that satisfies the axiom of commutativity. That is, for all* $a, b \in \mathbb{G}$*, we have* $a \cdot b = b \cdot a$*.*

Proposition 2.1. *Every cyclic group is abelian.*

An example of an important group used in cryptography is the group $\mathbb{Z}_n^*$ under the operation of multiplication modulo n. This group has order $\phi(n)$. Moreover, if n is prime, then $\mathbb{Z}_n^*$ is cyclic with order $\phi(n) = n - 1$.

Theorem 2.3 (Chinese remainder theorem (CRT))**.** *Let the integers* $\{n_i\}_{i \in [\lambda]}$ *be pairwise relatively prime and let* $\{a_i\}_{i \in [\lambda]}$ *be arbitrary integers. Then the system of simultaneous congruences*

$$a = a_i \pmod{n_i} \ (i = 1, \ldots, \lambda)$$

has a unique solution modulo $n = \prod_{i=1}^{\lambda} n_i$*.*

Let $n_i^* = n/n_i$. Now define $t_i = (n_i^*)^{-1} \pmod{n_i}$ and $e_i = n_i^* t_i$. (Note that since $\gcd(n_i, n_i^*) = 1$, we can compute t_i.) We see that $e_i = 1 \pmod{n_i}$, while for $j \neq i$, $n_i \mid n_j^*$ and so $e_j = 0 \pmod{n_i}$. Thus, we can obtain the unique solution modulo n as

$$a = \sum_{i=1}^{\lambda} a_i e_i \pmod{n}.$$

Definition 2.10 (Quadratic residues)**.** *Let $a \in \mathbb{Z}_n^*$. The element a is said to be a* quadratic residue *modulo n if it is congruent to a square modulo n, i.e. if there is an $x \in \mathbb{Z}_n^*$ such that $x^2 = a \pmod{n}$. (The element x is called a* square root *of a modulo n.) If no such x exists, a is called a* quadratic non-residue *modulo n. We denote the set of all quadratic residues modulo n by $\mathbb{QR}_n$ and the set of quadratic non-residues by $\overline{\mathbb{QR}}_n$.*

Note that when n is prime, half of the elements of $\mathbb{Z}_n^*$ are quadratic residues and the other half quadratic non-residues. Hence, $|\mathbb{QR}_n| = |\overline{\mathbb{QR}}_n| = \phi(n)/2 = (n-1)/2$. Furthermore, for n prime, we can determine whether an element $a \in \mathbb{Z}_n^*$ is a quadratic residue modulo n by using *Euler's criterion.*

Theorem 2.4 (Euler's criterion)**.** *Let p be an odd prime and a an integer such that* $\gcd(a, p) = 1$. *Then*

$$a^{\frac{p-1}{2}} = \begin{cases} 1 \pmod{p} & \text{if there is } x \in \mathbb{Z} \text{ such that } x^2 = a \pmod{p}, \\ -1 \pmod{p} & \text{if there is no such integer.} \end{cases}$$

In order to simplify the notation of the computation for Euler's criterion, we use the symbol $\left(\frac{a}{p}\right)$, introduded by Adrien-Marie Legendre. This symbol, called the *Legendre symbol*, comes with a number of useful properties which can be used to speed up the calculations for determining whether an integer is a quadratic-residue or a quadratic non-residue modulo a prime number.

Definition 2.11 (The Legendre symbol)**.** *Let p be an odd prime and a an integer. The* Legendre symbol $\left(\frac{a}{p}\right)$ *is defined as*

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{if } p \mid a, \\ 1 & \text{if } a \in \mathbb{QR}_p, \\ -1 & \text{if } a \in \overline{\mathbb{QR}}_p. \end{cases}$$

Note that by Definition 2.10, $0 \notin \mathbb{Z}_p^*$ and so $0 \notin \mathbb{QR}_p$ and $0 \notin \overline{\mathbb{QR}}_p$.

A useful generalization of the Legendre symbol to odd composite numbers is the *Jacobi symbol*. Like the Legendre symbol, the Jacobi symbol also comes with useful properties, which we omit here.

Definition 2.12 (The Jacobi symbol)**.** *Let n be an odd integer with prime factorization $n = {p_1}^{e_1}{p_2}^{e_2}\dots{p_\lambda}^{e_\lambda}$ and a an integer. The* Jacobi symbol $\left(\frac{a}{n}\right)$ *is defined as*

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{e_1}\left(\frac{a}{p_2}\right)^{e_2}\dots\left(\frac{a}{p_\lambda}\right)^{e_\lambda}.$$

We stress that $\left(\frac{a}{n}\right)$ can be efficiently computed even without knowing the prime factorization of n. This can be done by using properties of the Jacobi symbol. By $\mathbb{J}_n$ we denote the subgroup of all elements of $\mathbb{Z}_n^*$ having Jacobi symbol 1.

An important observation is that when n is a product of two distinct odd primes, p and q, then an element $a \in \mathbb{Z}_n^*$ is a quadratic residue modulo n if and only if $a \in \mathbb{QR}_p$ and $a \in \mathbb{QR}_q$. It follows that $|\mathbb{QR}_n| = |\mathbb{QR}_p||\mathbb{QR}_q| = (p-1)(q-1)/4 = \phi(n)/4$ and $|\overline{\mathbb{QR}}_n| = 3\phi(n)/4$. Also, note that a has Jacobi symbol 1 if and only if $\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = 1$ or $\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = -1$. This means that $|\mathbb{J}_n| = \phi(n)/2$.

Definition 2.13 (Blum integers and safe primes)**.** *A composite integer of the form $N = PQ$ is called a* Blum integer *if P and Q are distinct primes, each congruent to 3 modulo 4. A special type of Blum integer is when both $(P-1)/2$ and $(Q-1)/2$ are primes. In this case P and Q are called* safe primes.

For a Blum integer N, and an element $a \in \mathbb{QR}_N$, a has precisely four square roots modulo N. Moreover, exactly one of these square roots is in $\mathbb{QR}_N$.

Let us now recall the definition of the *group of signed quadratic residues* $\mathbb{QR}_N^+$ from [87] (see also [60, 80]).

Definition 2.14 (Signed quadratic residues)**.** *Let $N = PQ$ be a k-bit Blum integer. The group of* signed quadratic residues*, $\mathbb{QR}_N^+$, is defined as the group $\mathbb{QR}_N^+ = \{|x| : x \in \mathbb{QR}_N\}$, where $|x|$ is the absolute value when representing elements of $\mathbb{Z}_N$ as the set $\{-(N-1)/2, \dots, (N-1)/2\}$. For $g, h \in \mathbb{QR}_N^+$, the group operation is $g \circ h = |g \cdot h \pmod N|$.*

For simplicity of notation we will omit the group operation "$\circ$" when it is obvious by context which group operation is being used. As an example of the group $\mathbb{QR}_N^+$, let $P = 3$ and $Q = 7$. Then $\mathbb{Z}_N^* = \{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$ and $\mathbb{QR}_N = \{1, 4, 16\}$. If we represent the elements of $\mathbb{Z}_N$ as the set $\{-(N-1)/2, \ldots, (N-1)/2\}$, we have $\mathbb{QR}_N = \{1, 4, -5\}$. Thus, our set of signed quadratic residues modulo $N = PQ$ is $\mathbb{QR}_N^+ = \{1, 4, 5\}$.

Theorem 2.5 ([87], Lemma 1)**.** *Let N be a Blum integer. Then $(\mathbb{QR}_N^+, \circ)$ is a group of order $\phi(N)/4$ and is efficiently recognizable given only N, since $\mathbb{QR}_N^+ = \mathbb{J}_N^+ = \mathbb{J}_N \cap [(N-1)/2]$, where $\mathbb{J}_N^+$ denotes $\{|x| : x \in \mathbb{J}_N\}$. Moreover, if $\mathbb{QR}_N$ is cyclic, so is $\mathbb{QR}_N^+$.*

Ensuring that $\mathbb{J}_N$ is cyclic. In order to ensure that the group $\mathbb{J}_N$ is cyclic, and consequently $\mathbb{QR}_N$ is cyclic, we will require henceforth that all prime factors of $\phi(N)/4$ are pairwise distinct. Note that when $N = PQ$ is a Blum integer we can write $P = 2p' + 1$ and $Q = 2q' + 1$, where p' and q' are odd integers. Under the assumption that all prime factors of $\phi(N)/4$ are pairwise distinct, this implies that $\phi(N)/4 = p'q'$, which is odd, and $\gcd(P-1, Q-1) = \gcd(2p', 2q') = 2$. Now consider generators g_P and g_Q of the cyclic groups $\mathbb{Z}_P^*$ and $\mathbb{Z}_Q^*$, respectively. By the Chinese Remainder Theorem, since g_P has order $P-1$ in $\mathbb{Z}_P^*$ and g_Q has order $Q-1$ in $\mathbb{Z}_Q^*$, the unique element $g \in \mathbb{Z}_N^*$ with $g_P = g \pmod P$ and $g_Q = g \pmod Q$ has order $\operatorname{lcm}(P-1, Q-1) = (P-1)(Q-1)/\gcd(P-1, Q-1) = (P-1)(Q-1)/2 = 2p'q'$ in $\mathbb{Z}_N^*$. Furthermore, it is easy to see that $\left(\frac{g_P}{P}\right) = \left(\frac{g_Q}{Q}\right) = -1$. This means that $\left(\frac{g}{P}\right) = \left(\frac{g}{Q}\right) = -1$ and $\left(\frac{g}{N}\right) = 1$. Thus, $g \in \mathbb{J}_N$. As $\operatorname{ord}(g) = 2p'q' = \phi(N)/2 = |\mathbb{J}_N|$, g is a generator of $\mathbb{J}_N$.

Sampling a generator of $\mathbb{QR}_N$. Let $N = PQ$ be a k–bit Blum integer. As before we write $P = 2p' + 1$ and $Q = 2q' + 1$, for odd primes p' and q'. Assume that the prime factors of $\phi(N)/4$ are pairwise distinct and additionally that they are at least k'-bit integers, where $k' = \delta k$ for some fixed $0 \leq \delta < 1/2$. We know that $|\mathbb{QR}_N| = \phi(N)/4 = p'q'$. A random generator g of $\mathbb{QR}_N$ can be obtained, with overwhelming probability, by simply squaring a random element in $\mathbb{Z}_N^*$. We justify this as follows. Note that any generator g has four square roots in $\mathbb{Z}_N^*$ and has order $\operatorname{ord}(g) = p'q'$. Furthermore, the number of generators of $\mathbb{QR}_N$ is $\phi(p'q')$. Thus, the probability of picking a random element from $\mathbb{Z}_N^*$ such that its

square is a generator of $\mathbb{QR}_N$ is $4\phi(p'q')/|\mathbb{Z}_N^*| = 4\phi(p'q')/4p'q' = 1 - \mathcal{O}(2^{-k'})$. The distribution of generators of $\mathbb{QR}_N$ obtained this way is statistically close to uniform. Furthermore, because $\mathbb{QR}_N$ is cyclic, $|g|$ is a generator of $\mathbb{QR}_N^+$.

Generating random elements in $\mathbb{QR}_N^+$. In some of the constructions in this thesis we will need to generate random elements in the group $\mathbb{QR}_N^+$. It is obvious how to do this when we know the order of $\mathbb{QR}_N^+$, $\phi(N)/4$ (i.e. we know the factorization of N): uniformly select an integer $a \in \{1, \ldots, \phi(N)/4\}$ and set $A = g^a \pmod N$, where g is a generator of $\mathbb{QR}_N^+$. However, when the factorization of N is not known, we have to use an approximation: choose $a \in \{1, \ldots, \lfloor N/4 \rfloor\}$ and set $A = g^a \pmod N$. Note that if the prime factors of $\phi(N)/4$ have approximately the same size, then the uniform distributions over $\{1, \ldots, \phi(N)/4\}$ and $\{1, \ldots, \lfloor N/4 \rfloor\}$ are statistically close. The same argument applies for the generation of random elements in $\mathbb{QR}_N$.

2.1.3 Bilinear Maps

Bilinear maps have been widely used in the construction of cryptographic schemes since the work of Joux [93], where he used bilinear maps to construct a one-round tripartite key exchange scheme. Two other main contributions to the rapid increase of interest in bilinear maps were the identity-based non-interactive key exchange (ID-NIKE) scheme by Sakai, Ohgishi and Kasahara [122], and the identity-based encryption (IBE) scheme by Boneh and Franklin [32]. When using bilinear maps to construct cryptographic schemes, a commonly used approach is to treat them as "black-boxes". In this approach, the mathematical details of how the bilinear maps are selected or implemented are ignored; bilinear maps are treated as abstract mappings. This is the approach used in this thesis. We now give a basic introduction to this topic.

Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be three (multiplicatively-written) cyclic groups of the same order p. A bilinear map from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$ is a function

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

satisfying a special property:

Bilinearity. $\forall g, g' \in \mathbb{G}_1, h, h' \in \mathbb{G}_2$, we have

$$e(g \cdot g', h) = e(g, h) \cdot e(g', h) \text{ and } e(g, h \cdot h') = e(g, h) \cdot e(g, h').$$

Consequently, for any $a, b \in \mathbb{Z}$: $e(g^a, h^b) = e(g, h)^{ab} = e(g^b, h^a)$.

Bilinear maps are also called *pairings*, as they associate pairs of elements in $\mathbb{G}_1 \times \mathbb{G}_2$ with an element in $\mathbb{G}_T$. To be of practical use in cryptography, we also require the map to satisfy two additional properties:

Non-Degeneracy. For $\mathbb{G}_1 = \langle g_1 \rangle, \mathbb{G}_2 = \langle g_2 \rangle$, we have that $\mathbb{G}_T = \langle e(g_1, g_2) \rangle$.

Efficiently Computable. There exists an efficient algorithm that computes the map e for any pair of inputs in $\mathbb{G}_1 \times \mathbb{G}_2$.

We say the map e is an *admissible bilinear map* if it satisfies the three aforementioned properties. Henceforth we implicitly mean admissible bilinear map when we say bilinear map or pairing.

Pairings can be classified into four different types, based on the concrete structures of the underlying groups [44, 68].

Type 1. $\mathbb{G}_1 = \mathbb{G}_2$ (symmetric pairings).

Type 2. $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is an efficiently-computable homomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$. In this situation there is no known way to efficiently hash bit strings into $\mathbb{G}_2$.

Type 3. $\mathbb{G}_1 \neq \mathbb{G}_2$ and no efficiently-computable homomorphism is known between $\mathbb{G}_1$ and $\mathbb{G}_2$. However here, it is usually possible to hash into $\mathbb{G}_2$.

Type 4. $\mathbb{G}_1 \neq \mathbb{G}_2$ (exception). This is a new type of pairings where $\mathbb{G}_2$ is a non-cyclic group and it is possible to hash into $\mathbb{G}_2$. Furthermore, there is an efficiently-computable homomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$.

Note that for none of Type 2, Type 3 or Type 4 pairings (asymmetric pairings) does there exist an efficiently-computable homomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$. The

situation where $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is an efficiently-computable homomorphism in both directions ($\mathbb{G}_2 \rightarrow \mathbb{G}_1$ and $\mathbb{G}_1 \rightarrow \mathbb{G}_2$) can also be interpreted as Type 1.

The use of symmetric pairings in cryptographic schemes allows relatively simpler description of the schemes and of their security proofs, when compared to the use of asymmetric pairings. However, as indicated by [68], Type 1 pairings are expected to be less efficient as the security parameter grows. Type 2 and Type 3 pairings seem to be better choices if we take into consideration efficiency of their implementations. Type 3 is more efficient than Type 2 since the former provides faster pairing computation, shorter representation for elements in $\mathbb{G}_2$, less complex group operations in $\mathbb{G}_2$ and a more efficient membership test in $\mathbb{G}_2$. Nevertheless, as mentioned before, no efficiently-computable homomorphism is known between $\mathbb{G}_1$ and $\mathbb{G}_2$ for Type 3 pairings. Thus, for schemes that require an homomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ in the scheme itself or in its security proof, Type 2 pairings seem to be the best choice. Type 4 pairings, introduced in [125], are an option for schemes that require both hashing into $\mathbb{G}_2$ and the homomorphism ψ. However, the hashing into $\mathbb{G}_2$ is not so cheap. (Though in [43] a new and cheaper method for hashing into $\mathbb{G}_2$ is proposed.) More importantly, with some small probability, the pairing might not satisfy the non-degeneracy property.

Known examples of pairings are the Weil and Tate pairings [66, 139], which are derived from elliptic curves over finite fields. An elliptic curve is a set of pairs (x, y) satisfying a finite field equation of the form $y^2 = x^3 + ax + b$, where a and b are parameters of the curve. *Supersingular* elliptic curves are a special class of elliptic curves with some additional algebraic structure. Symmetric pairings are derived from supersingular elliptic curves whereas asymmetric pairings can be derived from ordinary elliptic curves. For further details on pairings and elliptic curves we refer to [67, 68].

Pairing Parameter Generator. A *Type i pairing parameter generator*, denoted by $\mathcal{G}_i$, is a polynomial time algorithm that on input a security parameter 1^k, returns the description of three (multiplicatively-written) cyclic groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ of the same order p, generators g_1, g_2 (for $\mathbb{G}_1, \mathbb{G}_2$, respectively), and an admissible pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Depending on the type of pairing (1, 2, 3 or 4), $\mathcal{G}_i(1^k)$ also outputs the description of an efficiently computable homomorphism $\psi : \mathbb{G}_2 \rightarrow$

$\mathbb{G}_1$, with $g_1 = \psi(g_2)$. We write $\mathcal{PG}_i = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, p, e, \psi) \leftarrow \mathcal{G}_i(1^k)$. For symmetric pairings (Type 1 pairings) $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ we can simply write $\mathcal{PG} = (\mathbb{G}, \mathbb{G}_T, g, p, e) \leftarrow \mathcal{G}(1^k)$, where $\mathbb{G}$ is a group of order p generated by g.

2.1.4 Multilinear Maps

Multilinear maps and their applications to cryptography were first put forward by Boneh and Silverberg in [33]. They studied the problem of finding efficiently computable, non-degenerate ℓ-linear maps $e : \mathbb{G}_1^\ell \to \mathbb{G}_T$, where $\mathbb{G}_1$ and $\mathbb{G}_T$ are groups of the same prime order and computing the discrete logarithm in $\mathbb{G}_1$ is hard. In [33] the authors motivated the construction of such multilinear maps by discussing several potential applications to cryptography, such as one-round $(\ell + 1)$-user key exchange schemes (an extension of Joux's scheme [94]) and efficient broadcast encryption schemes. Asymmetric multilinear maps $e : \mathbb{G}_1 \times \ldots \times \mathbb{G}_\ell \to \mathbb{G}_T$, for different groups $\mathbb{G}_i$, were considered by Rothblum in [121], where he considered the problem of circular security of bit encryption.

In this thesis our framework of multilinear maps is defined so that a user can run a group generator $\mathcal{MG}_\ell(1^k)$ to obtain a sequence of leveled multilinear groups $\mathbb{G}_1, \ldots, \mathbb{G}_\ell$ each of prime order p and a set of bilinear maps $\{e_{i,j} : \mathbb{G}_i \times \mathbb{G}_j \to \mathbb{G}_{i+j} \mid i, j \geq 1, i + j \leq \ell\}$. This can be seen as implementing multilinear maps. We give more details next.

Multilinear maps. An ℓ-group system consists of ℓ cyclic groups $\mathbb{G}_1, \ldots, \mathbb{G}_\ell$ of prime order p, along with bilinear maps $e_{i,j} : \mathbb{G}_i \times \mathbb{G}_j \to \mathbb{G}_{i+j}$ for all $i, j \geq 1$ with $i+j \leq \ell$. Let g_i be a canonical generator of $\mathbb{G}_i$ (included in the group's description). The map $e_{i,j}$ satisfies

$$e_{i,j}(g_i^a, g_j^b) = g_{i+j}^{ab} \ : \ \forall a, b \in \mathbb{Z}_p.$$

When i, j are clear, we will simply write e instead of $e_{i,j}$. It will also be convenient to abbreviate $e(h_1, \ldots, h_j) := e(h_1, e(h_2, \ldots, e(h_{j-1}, h_j) \ldots))$ for $h_j \in \mathbb{G}_{i_j}$ and $i = (i_1 + i_2 + \ldots + i_j) \leq \ell$. By induction, it is easy to see that this map is j-linear. Additionally, we define $e(g) := g$. Finally, it can also be useful to define the group $\mathbb{G}_0 = \mathbb{Z}_{|\mathbb{G}_1|}$ of exponents to which this pairing family naturally extends; then, we also consider maps $e_{0,j}$ and $e_{i,0}$, for $i, j \leq \ell$. We will assume that ℓ-group systems

$\mathcal{MPG}_\ell = \{\{\mathbb{G}_i\}_{i\in[\ell]}, p, \{e_{i,j}\}_{i,j\geq 1, i+j\leq \ell}\}$ can be generated by a *multilinear maps parameter generator* $\mathcal{MG}_\ell$, on input a security parameter 1^k.

The GGH candidate. We currently do not have multilinear maps between groups with cryptographically hard problems. However, as we will see in Section 4.2.1, Garg, Gentry, and Halevi [72] (henceforth GGH) suggest a concrete candidate for an "approximation" of multilinear maps, named *graded encoding systems*. For this chapter, we consider only the framework of multilinear maps described above.

2.1.5 Computational Hardness Assumptions

We describe here the computational assumptions which form the basis of security of the cryptographic schemes presented in this thesis. For completeness, we also include some basic related assumptions.

Number Theoretic Assumptions

Factoring Assumption. The most established hardness assumption is the *factoring assumption*. Informally, it states that given a composite integer N, it is infeasible to find its factorization $N = pq$, where p and q are distinct large primes. The difficulty of finding the prime factors will depend on properties of these numbers, like size or special form. Thus, to formalize the factoring assumption, we need to consider an instance generator of the composite number N.

Definition 2.15. *Let $n(k)$ be a function of the security parameter k. We define* RSAgen *as a polynomial-time algorithm that on input 1^k outputs (N, p, q) such that $N = pq$ is an n-bit integer, with p and q distinct primes, possibly with additional constraints (to be specified).*

Definition 2.16 (Factoring assumption)**.** *For an algorithm $\mathcal{A}$ we define its factoring advantage as*

$$\mathrm{Adv}^{\mathrm{fac}}_{\mathcal{A},\mathsf{RSAgen}}(k) = \Pr[\{p, q\} \leftarrow \mathcal{A}(N) : (N, p, q) \leftarrow \mathsf{RSAgen}(1^k)].$$

The factoring assumption with respect to RSAgen *is that* $\mathrm{Adv}^{\mathrm{fac}}_{\mathcal{A},\mathsf{RSAgen}}(k)$ *is negligible for all probabilistic polynomial-time (PPT) algorithms $\mathcal{A}$.*

The Quadratic Residuosity (QR) Assumption. This assumption states that given an odd composite integer N and a uniformly random element $a \in \mathbb{J}_N$, it is infeasible to decide whether or not $a \in \mathbb{QR}_N$. The QR assumption is believed to be valid, for example, when N is a Blum integer. In other words, for Blum integers N, the group $\mathbb{QR}_N$ is believed not to be efficiently reconizable. Another important fact is that computing square roots in this group is equivalent to factoring the modulus N, as proved by Rabin in [114].

The Discrete Logarithm (DL) Assumption. The security of many cryptographic constructions relies on hardness assumptions related to the discrete logarithm assumption. Let $\mathbb{G} = \langle g \rangle$ be a cyclic group. The DL problem is: given $h \in \mathbb{G}$, find the discrete logarithm of h to the base g, denoted $\mathrm{dlog}_g h$. The DL assumption states that it is infeasible to solve the DL problem. Formally, we consider a discrete logarithm parameter generator DLgen.

Definition 2.17. *We define a* discrete logarithm parameter generator *as a polynomial-time algorithm* DLgen *which on input a security parameter* 1^k*, generates an integer* p *along with the description of a cyclic group* $\mathbb{G} = \langle g \rangle$ *of order* p*.*

Definition 2.18 (The DL assumption)**.** *Consider the following experiment associated with algorithm* $\mathcal{A}$*.*

$$
\begin{array}{l}
\text{Experiment } \mathrm{Exp}^{\mathrm{DL}}_{\mathcal{A},\mathsf{DLgen}}(1^k) \\
\quad (\mathbb{G}, g, p) \leftarrow \mathsf{DLgen}(1^k) \\
\quad a \leftarrow \mathbb{Z}_p \\
\quad a' \leftarrow \mathcal{A}(1^k, \mathbb{G}, p, g, g^a) \\
\quad \textit{If } a = a' \textit{ then return 1 else return 0}
\end{array}
$$

The advantage of $\mathcal{A}$ *in the above experiment is defined as*

$$\mathrm{Adv}^{\mathrm{DL}}_{\mathcal{A},\mathsf{DLgen}}(k) = \left| \Pr\left[\mathrm{Exp}^{\mathrm{DL}}_{\mathcal{A},\mathsf{DLgen}}(1^k) = 1 \right] \right|.$$

The DL assumption with respect to DLgen *is that* $\mathrm{Adv}^{\mathrm{DL}}_{\mathcal{A},\mathsf{DLgen}}(k)$ *is negligible for all PPT algorithms* $\mathcal{A}$*.*

Some examples of groups in which the DL problem is believed to be intractable are: $\mathbb{Z}_p^*$ for some large prime p, where $p-1$ has at least one large prime factor; cyclic subgroups $\mathbb{H} \subset \mathbb{Z}_p^*$ of prime order q; some elliptic curve groups.

The Computational Diffie-Hellman (CDH) Assumption. The CDH problem with respect to DLgen is: given a random instance (g, g^a, g^b), for $a, b \leftarrow \mathbb{Z}_p$,

compute g^{ab}. The CDH assumption states that it is infeasible to solve the CDH problem.

Definition 2.19 (The CDH assumption)**.** *Consider the following experiment associated with algorithm $\mathcal{A}$.*

$$\begin{array}{l}\text{Experiment } \mathrm{Exp}^{\mathrm{CDH}}_{\mathcal{A},\mathsf{DLgen}}(1^k)\\ \quad (\mathbb{G}, g, p) \leftarrow \mathsf{DLgen}(1^k)\\ \quad a, b \leftarrow \mathbb{Z}_p\\ \quad Z \leftarrow \mathcal{A}(1^k, \mathbb{G}, p, g, g^a, g^b)\\ \quad \textit{If } Z = g^{ab} \textit{ then return 1 else return 0}\end{array}$$

The advantage of $\mathcal{A}$ in the above experiment is defined as

$$\mathrm{Adv}^{\mathrm{CDH}}_{\mathcal{A},\mathsf{DLgen}}(k) = \left|\Pr\left[\mathrm{Exp}^{\mathrm{CDH}}_{\mathcal{A},\mathsf{DLgen}}(1^k) = 1\right]\right|.$$

The CDH assumption with respect to DLgen *is that* $\mathrm{Adv}^{\mathrm{CDH}}_{\mathcal{A},\mathsf{DLgen}}(k)$ *is negligible for all PPT algorithms $\mathcal{A}$.*

This assumption is closely related to the DL assumption; if the DL assumption does not hold, the CDH problem can be efficiently solved as follows: given (g, g^a, g^b), solve the DL problem for g^a, finding a, and then compute $(g^b)^a$. The CDH assumption only states that g^{ab} cannot be efficiently computed, but this does not mean that it is also infeasible to get some information about g^{ab} (e.g. its most significant bit), given the problem instance.

Shmuely [128] and McCurley [106] proved that over the group of quadratic residues $\mathbb{QR}_N$, where $N = PQ$ is the product of two large primes, the CDH problem is at least as hard as factoring N.

The Decisional Diffie-Hellman (DDH) Assumption. This is a stronger assumption than the CDH assumption. Roughly, it states that given two distributions (g, g^a, g^b, g^{ab}) and (g, g^a, g^b, g^c), for $a, b, c \leftarrow \mathbb{Z}_p$, it is infeasible for any algorithm to tell them apart. This means that now it is infeasible to get any information about g^{ab}, given (g, g^a, g^b).

Definition 2.20 (The DDH assumption)**.** *Consider the following experiment asso-*

ciated with algorithm $\mathcal{A}$.

$$\begin{array}{l}\text{Experiment } \mathrm{Exp}^{\mathrm{DDH},\beta}_{\mathcal{A},\mathsf{DLgen}}(1^k)\\ \quad (\mathbb{G}, g, p) \leftarrow \mathsf{DLgen}(1^k)\\ \quad a, b, c \leftarrow \mathbb{Z}_p\\ \quad \textit{If } \beta = 1 \textit{ then } T = g^{ab} \textit{ else } T = g^c\\ \quad \beta' \leftarrow \mathcal{A}(1^k, \mathbb{G}, p, g, g^a, g^b, T)\\ \quad \textit{Return } \beta'\end{array}$$

The advantage of $\mathcal{A}$ in the above experiment is defined as

$$\mathrm{Adv}^{\mathrm{DDH}}_{\mathcal{A},\mathsf{DLgen}}(k) = 2\left|\Pr\left[\mathrm{Exp}^{\mathrm{DDH},\beta}_{\mathcal{A},\mathsf{DLgen}}(1^k) = \beta : \beta \leftarrow \{0,1\}\right] - 1/2\right|,$$

which can also be written as

$$\mathrm{Adv}^{\mathrm{DDH}}_{\mathcal{A},\mathsf{DLgen}}(k) = \left|\Pr\left[\mathrm{Exp}^{\mathrm{DDH},1}_{\mathcal{A},\mathsf{DLgen}}(1^k) = 1\right] - \Pr\left[\mathrm{Exp}^{\mathrm{DDH},0}_{\mathcal{A},\mathsf{DLgen}}(1^k) = 1\right]\right|.$$

The DDH assumption with respect to DLgen *is that* $\mathrm{Adv}^{\mathrm{DDH}}_{\mathcal{A},\mathsf{DLgen}}(k)$ *is negligible for all PPT algorithms $\mathcal{A}$.*

It is easy to see that solving the CDH problem enables one to solve the DDH problem. However, there are groups for which the DDH problem is easy while the CDH problem is believed to be hard. Groups for which solving the distinguishability problem is easy but solving the computational problem is hard are called *gap groups*.

Note for example, that the DDH problem is easy in the group $\mathbb{Z}_p^*$, where p is a prime: given g^a, g^b, T, a PPT algorithm $\mathcal{A}$ can compute $\left(\frac{g^a}{p}\right)$ and $\left(\frac{g^b}{p}\right)$ and then predict $\left(\frac{g^{ab}}{p}\right) = s$. Now $\mathcal{A}$ checks whether or not $\left(\frac{T}{p}\right) = s$. If so it outputs 1, otherwise 0. The advantage of $\mathcal{A}$ in solving the DDH problem is $|1 - 1/2| = 1/2$. However, DDH is believed to be intractable, e.g. in subgroups of $\mathbb{Z}_p^*$ of prime order (such as the subgroup of quadratic residues $\mathbb{QR}_p$ in $\mathbb{Z}_p^*$ for a safe prime p); in the cyclic subgroup $\mathbb{QR}_N$ in $\mathbb{Z}_N^*$, where N is a product of safe primes and its factorization is unknown; in some elliptic curve groups.

The Strong Diffie-Hellman (SDH) Assumption. The SDH assumption [1] is that there is no PPT algorithm having non-negligible advantage in solving the CDH problem on input (g, A, B) when given access to a DDH oracle for fixed g and A, denoted by $\mathrm{DDH}_{g,A}(\cdot,\cdot)$. Here g is a randomly selected generator of a cyclic group

$\mathbb{G}$ (with order not necessarily known), A and B are uniformly selected from $\mathbb{G}$, and the solution to the CDH problem is defined as $g^{(\mathrm{dlog}_g A)(\mathrm{dlog}_g B)}$. The DDH oracle $\mathrm{DDH}_{g,A}(\hat{B}, \hat{C})$ returns 1 if $\hat{B}^{\mathrm{dlog}_g A} = \hat{C}$ and 0 otherwise, where $(\hat{B}, \hat{C}) \in \mathbb{G} \times \mathbb{G}$.

We consider in more detail the definition of the SDH assumption for the group of signed quadratic residues $\mathbb{QR}_N^+$. Since the group is efficiently recognizable, we do not permit access to the oracle $\mathrm{DDH}_{g,A}(\hat{B}, \hat{C})$ for $(\hat{B}, \hat{C}) \notin \mathbb{QR}_N^+ \times \mathbb{QR}_N^+$. We specify an instance generator RSAgen as follows.

Let $n(k)$ be a function and δ a constant with $0 \leq \delta < 1/2$. $\mathsf{RSAgen}(1^k)$ generates elements (N, P, Q) such that $N = PQ$ is an n-bit Blum integer and all prime factors of $\phi(N)/4$ are pairwise distinct and have at least δn bits. As we saw in Section 2.1.2, these conditions ensure that $\mathbb{J}_N$ is cyclic and that the square g of a random element in $\mathbb{Z}_N^*$ generates $\mathbb{QR}_N$ (i.e. $\langle g \rangle = \mathbb{QR}_N$) with high probability $1 - \mathcal{O}(2^{-\delta n(k)})$.

Definition 2.21 (The SDH assumption)**.** *Consider the following experiment associated with algorithm $\mathcal{A}$.*

$$
\begin{array}{l}
\text{Experiment } \mathrm{Exp}^{\mathrm{SDH}}_{\mathcal{A},\mathsf{RSAgen}}(1^k) \\
\quad (N, P, Q) \leftarrow \mathsf{RSAgen}(1^k) \\
\quad g \leftarrow \mathbb{QR}_N^+, \text{ where } \langle g \rangle = \mathbb{QR}_N^+ \\
\quad A, B \leftarrow \mathbb{QR}_N^+ \\
\quad Z \leftarrow \mathcal{A}^{\mathrm{DDH}_{g,A}(\cdot,\cdot)}(1^k, N, g, A, B) \\
\quad \text{If } Z = g^{(\mathrm{dlog}_g A)(\mathrm{dlog}_g B)} \text{ then return 1 else return 0}
\end{array}
$$

The advantage of $\mathcal{A}$ in the above experiment is defined as

$$\mathrm{Adv}^{\mathrm{SDH}}_{\mathcal{A},\mathsf{RSAgen}}(k) = \left| \Pr\left[\mathrm{Exp}^{\mathrm{SDH}}_{\mathcal{A},\mathsf{RSAgen}}(1^k) = 1 \right] \right|.$$

The SDH assumption with respect to RSAgen is that $\mathrm{Adv}^{\mathrm{SDH}}_{\mathcal{A},\mathsf{RSAgen}}(k)$ is negligible for all PPT algorithms $\mathcal{A}$.

Theorem 2.6 (Breaking SDH $\Rightarrow$ Factoring [87])**.** *If the factoring assumption holds relative to RSAgen, then the SDH assumption holds in the group of signed quadratic residues $\mathbb{QR}_N^+$ relative to RSAgen. In particular for every algorithm $\mathcal{A}$ solving the SDH problem, there exists a factoring algorithm $\mathcal{B}$ (with roughly the same running time as $\mathcal{A}$) such that*

$$\mathrm{Adv}^{\mathrm{SDH}}_{\mathcal{A},\mathsf{RSAgen}}(k) \leq \mathrm{Adv}^{\mathrm{fac}}_{\mathcal{B},\mathsf{RSAgen}}(k) + \mathcal{O}(2^{-\delta n(k)}).$$

Note that the proof of Theorem 2.6 is not valid in the group of quadratic residues $\mathbb{QR}_N$. This is because in the reduction, the factoring algorithm $\mathcal{B}$ has to simulate a DDH oracle and hence has to be able to determine membership in $\mathbb{QR}_N$, which is believed to be infeasible in this group.

The Double-Strong Diffie-Hellman (DSDH) Assumption. We define a variant of the SDH assumption, where instead of having one DDH oracle, $\mathrm{DDH}_{g,A}(\cdot,\cdot)$ (for fixed g, A), we also have a second DDH oracle, $\mathrm{DDH}_{g,B}(\cdot,\cdot)$ (for fixed g, B).

Theorem 2.7 (Breaking DSDH $\Rightarrow$ Factoring)**.** *If the factoring assumption holds relative to* RSAgen*, then the DSDH assumption also holds in the group of signed quadratic residues $\mathbb{QR}_N^+$ relative to* RSAgen*. In particular, for every algorithm $\mathcal{A}$ solving the DSDH problem, there exists a factoring algorithm $\mathcal{B}$ (with roughly the same running time as $\mathcal{A}$) such that*

$$\mathrm{Adv}^{\mathrm{DSDH}}_{\mathcal{A},\mathsf{RSAgen}}(k) \leq \mathrm{Adv}^{\mathrm{fac}}_{\mathcal{B},\mathsf{RSAgen}}(k) + \mathcal{O}(2^{-\delta n(k)}).$$

Proof. The original proof of Theorem 2.6 in [87] shows how to handle a single DDH oracle $\mathrm{DDH}_{g,A}(\cdot,\cdot)$. By symmetry of the set-up used in the proof, the same procedure can also be used to (simultaneously) handle the oracle $\mathrm{DDH}_{g,B}(\cdot,\cdot)$. □

Hardness Assumptions based on Bilinear and Multilinear Maps

The Bilinear Diffie-Hellman (BDH) Assumption (Informal). Let $\mathcal{PG} = (\mathbb{G}, \mathbb{G}_T, g, p, e)$ be the output of a symmetric pairing parameter generator, $\mathcal{G}$. The BDH problem is as follows: Given (g, g^a, g^b, g^c) for $a, b, c \leftarrow \mathbb{Z}_p$ and $g \in \mathbb{G}$, compute $e(g,g)^{abc} \in \mathbb{G}_T$. The BDH assumption states that it is infeasible to solve the BDH problem.

The Decisional Bilinear Diffie-Hellman (DBDH) Assumption for Symmetric Pairings. The DBDH assumption is the decisional counterpart of the BDH assumption. It states that given two distributions $(g, g^a, g^b, g^c, e(g,g)^{abc})$ and $(g, g^a, g^b, g^c, e(g,g)^z)$, for $a, b, c, z \leftarrow \mathbb{Z}_p$ and $g \in \mathbb{G}$ it is infeasible for any algorithm to tell them apart.

Definition 2.22 (The DBDH assumption)**.** *Consider the following experiment associated with algorithm $\mathcal{A}$ and pairing parameter generator $\mathcal{G}$.*

$$\begin{array}{l}\text{Experiment } \mathrm{Exp}^{\mathrm{DBDH},\beta}_{\mathcal{A},\mathcal{G}}(1^k)\\ \quad \mathcal{PG} \leftarrow \mathcal{G}(1^k)\\ \quad a,b,c,z \leftarrow \mathbb{Z}_p\\ \quad \textit{If } \beta = 1 \textit{ then } T \leftarrow e(g,g)^{abc} \textit{ else } T \leftarrow e(g,g)^z\\ \quad \beta' \leftarrow \mathcal{A}(1^k, \mathcal{PG}, g^a, g^b, g^c, T)\\ \quad \textit{Return } \beta'\end{array}$$

The advantage of $\mathcal{A}$ in the above experiment is defined as

$$\mathrm{Adv}^{\mathrm{DBDH}}_{\mathcal{A},\mathcal{G}}(k) = 2\left|\Pr\left[\mathrm{Exp}^{\mathrm{DBDH},\beta}_{\mathcal{A},\mathcal{G}}(1^k) = \beta : \beta \leftarrow \{0,1\}\right] - 1/2\right|.$$

We say that the DBDH assumption relative to $\mathcal{G}$ holds if $\mathrm{Adv}^{\mathrm{DBDH}}_{\mathcal{A},\mathcal{G}}(k)$ is negligible for all PPT algorithms $\mathcal{A}$.

The Decisional Bilinear Diffie-Hellman Assumption for Type 2 Pairings (DBDH-2). Let $\mathcal{PG}_2 = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, p, e, \psi)$ be the output of a Type 2 pairing parameter generator, $\mathcal{G}_2$. In this thesis we consider the following version of the Decisional Bilinear Diffie-Hellman problem for Type 2 pairings, as introduced by Galindo in [69]: Given $(g_2, g_2^a, g_2^b, g_1^c, T) \in \mathbb{G}_2^3 \times \mathbb{G}_1 \times \mathbb{G}_T$ as input, the problem is to decide whether or not $T = e(g_1, g_2)^{abc}$.

Definition 2.23 (The DBDH-2 assumption)**.** *Consider the following experiment associated with algorithm $\mathcal{A}$ and pairing parameter generator $\mathcal{G}_2$.*

$$\begin{array}{l}\text{Experiment } \mathrm{Exp}^{\text{DBDH-2},\beta}_{\mathcal{A},\mathcal{G}_2}(1^k)\\ \quad \mathcal{PG}_2 \leftarrow \mathcal{G}_2(1^k)\\ \quad a,b,c,z \leftarrow \mathbb{Z}_p\\ \quad \textit{If } \beta = 1 \textit{ then } T \leftarrow e(g_1,g_2)^{abc} \textit{ else } T \leftarrow e(g_1,g_2)^z\\ \quad \beta' \leftarrow \mathcal{A}(1^k, \mathcal{PG}_2, g_2^a, g_2^b, g_1^c, T)\\ \quad \textit{Return } \beta'\end{array}$$

The advantage of $\mathcal{A}$ in the above experiment is defined as

$$\mathrm{Adv}^{\text{DBDH-2}}_{\mathcal{A},\mathcal{G}_2}(k) = 2\left|\Pr\left[\mathrm{Exp}^{\text{DBDH-2},\beta}_{\mathcal{A},\mathcal{G}_2}(1^k) = \beta : \beta \leftarrow \{0,1\}\right] - 1/2\right|.$$

We say that the DBDH-2 assumption relative to $\mathcal{G}_2$ holds if $\mathrm{Adv}^{\text{DBDH-2}}_{\mathcal{A},\mathcal{G}_2}(k)$ is negligible for all PPT algorithms $\mathcal{A}$.

Some implications of bilinear maps. The existence of the bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ as presented in Section 2.1.3 has direct implications for the groups $\mathbb{G}_1$ and $\mathbb{G}_2$:

- The MOV reduction [108] – given groups $\mathbb{G}_1$, $\mathbb{G}_2$ and a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, the DL problem in $\mathbb{G}_i$ ($i \in \{1, 2\}$) is not harder than the DL problem in $\mathbb{G}_T$. Suppose we have $g_1 \in \mathbb{G}_1$ and want to calculate the discrete logarithm of $g_1^a \in \mathbb{G}_1$. If $e(g_1, g_2) = g_T \in \mathbb{G}_T$ then $e(g_1^a, g_2) = e(g_1, g_2)^a = g_T^a$. So we can find a by calculating the discrete logarithm of $g_T^a \in \mathbb{G}_T$. The same argument is valid for calculating the discrete logarithm of $g_2^a \in \mathbb{G}_2$.
- Given a group $\mathbb{G}_1$ equipped with a Type 1 pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$, DDH is easy in $\mathbb{G}_1$ (pointed out by Joux and Nguyen in [95]). To see this note that given $(g_1, {g_1}^a, {g_1}^b, {g_1}^c) \in \mathbb{G}_1^4$ we have that $c = ab \pmod p$ if and only if $e({g_1}^a, {g_1}^b) = e(g_1, {g_1}^c)$.
- Given groups $\mathbb{G}_1, \mathbb{G}_2$ and a Type 2 pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, DDH is easy in $\mathbb{G}_2$. This follows because of the existence of the efficiently computable homomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$. Given $(g_2, {g_2}^a, {g_2}^b, T) \in \mathbb{G}_2^4$, one can decide whether $T = {g_2}^{ab}$ or $T = {g_2}^c$, for a uniform $c \leftarrow \mathbb{Z}_p$, by testing whether or not $e(\psi(T), g_2) = e(\psi(g_2^a), g_2^b)$.

The ℓ-Multilinear Decisional Diffie-Hellman (ℓ-MDDH) Assumption. Let $\mathcal{MPG}_\ell = \{\{\mathbb{G}_i\}_{i\in[\ell]}, p, \{e_{i,j}\}_{i,j\geq 1, i+j\leq \ell}\}$ be an ℓ-group system output by a multilinear maps parameter generator, $\mathcal{MG}_\ell$, on input a security parameter 1^k. The ℓ-MDDH assumption states that given $(g, g^{x_1}, \ldots, g^{x_{\ell+1}})$ (for $g \leftarrow \mathbb{G}_1$ and uniform exponents x_i), the element $e(g^{x_1}, \ldots, g^{x_\ell})^{x_{\ell+1}} \in \mathbb{G}_\ell$ is computationally indistinguishable from a uniform $\mathbb{G}_\ell$-element.

Definition 2.24 (The ℓ-MDDH assumption)**.** *Consider the following experiment associated with algorithm $\mathcal{A}$ and pairing parameter generator $\mathcal{MG}_\ell$.*

Experiment $\mathrm{Exp}^{\ell\text{-MDDH},\beta}_{\mathcal{A},\mathcal{MG}_\ell}(1^k)$
$\mathcal{MPG}_\ell \leftarrow \mathcal{MG}_\ell(1^k)$
$x_1, \ldots, x_{\ell+1} \leftarrow \mathbb{Z}_p$; $g \leftarrow \mathbb{G}_1$
If $\beta = 1$ *then* $T \leftarrow e(g^{x_1}, \ldots, g^{x_\ell})^{x_{\ell+1}} \in \mathbb{G}_\ell$ *else* $T \leftarrow \mathbb{G}_\ell$
$\beta' \leftarrow \mathcal{A}(1^k, \mathcal{MPG}_\ell, g, g^{x_1}, \ldots, g^{x_{\ell+1}}, T)$
Return β'

The advantage of $\mathcal{A}$ in the above experiment is defined as

$$\mathrm{Adv}^{\ell\text{-MDDH}}_{\mathcal{A},\mathcal{MG}_\ell}(k) = 2\left|\Pr\left[\mathrm{Exp}^{\ell\text{-MDDH},\beta}_{\mathcal{A},\mathcal{MG}_\ell}(1^k) = \beta : \beta \leftarrow \{0,1\}\right] - 1/2\right|.$$

We say that the ℓ-MDDH assumption relative to $\mathcal{MG}_\ell$ holds if $\mathrm{Adv}^{\ell\text{-MDDH}}_{\mathcal{A},\mathcal{MG}_\ell}(k)$ is negligible for all PPT algorithms $\mathcal{A}$.

The $(\ell+1)$-Power Assumption. Let $\mathcal{MPG}_\ell = \{\{\mathbb{G}_i\}_{i\in[\ell]}, p, \{e_{i,j}\}_{i,j\geq 1, i+j\leq \ell}\}$ be an ℓ-group system output by a multilinear maps parameter generator, $\mathcal{MG}_\ell$, on input a security parameter 1^k. The $(\ell+1)$-power assumption states that given (g, g^x) (for $g \leftarrow \mathbb{G}_1$ and uniform x), the element $e(\underbrace{g^x, \ldots, g^x}_{\ell \text{ times}})^x \in \mathbb{G}_\ell$ is computationally indistinguishable from a uniform $\mathbb{G}_\ell$-element.

Definition 2.25 (The $(\ell+1)$-power assumption)**.** *Consider the following experiment associated with algorithm $\mathcal{A}$ and pairing parameter generator $\mathcal{MG}_\ell$.*

Experiment $\mathrm{Exp}^{(\ell+1)\text{-power},\beta}_{\mathcal{A},\mathcal{MG}_\ell}(1^k)$
  $\mathcal{MPG}_\ell \leftarrow \mathcal{MG}_\ell(1^k)$
  $x \leftarrow \mathbb{Z}_p$; $g \leftarrow \mathbb{G}_1$
  If $\beta = 1$ *then* $T \leftarrow e(g^x, \ldots, g^x)^x \in \mathbb{G}_\ell$ *else* $T \leftarrow \mathbb{G}_\ell$
  $\beta' \leftarrow \mathcal{A}(1^k, \mathcal{MPG}_\ell, g, g^x, T)$
  Return β'

The advantage of $\mathcal{A}$ in the above experiment is defined as

$$\mathrm{Adv}^{(\ell+1)\text{-power}}_{\mathcal{A},\mathcal{MG}_\ell}(k) = 2\left|\Pr\left[\mathrm{Exp}^{(\ell+1)\text{-power},\beta}_{\mathcal{A},\mathcal{MG}_\ell}(1^k) = \beta : \beta \leftarrow \{0,1\}\right] - 1/2\right|.$$

We say that the $(\ell+1)$-power assumption relative to $\mathcal{MG}_\ell$ holds if $\mathrm{Adv}^{(\ell+1)\text{-power}}_{\mathcal{A},\mathcal{MG}_\ell}(k)$ is negligible for all PPT algorithms $\mathcal{A}$.

2.2 Provable Security

2.2.1 History

For many years, cryptographic schemes were designed in an *ad hoc* way. A cryptographic goal would be recognized and then a solution would be offered. The scheme

would be considered secure if no attacks against it were found. This means that cryptographic schemes offered very little security guarantees – security only against known attacks. If a new attack was found, the scheme would be either repaired or discarded. If repaired, the scheme would still be subject to possible unknown attacks.

In 1949, Claude Shannon published a paper entitled "Communication Theory of Secrecy Systems" [126], which had a great influence on the rigorous study of cryptography. He proved that the one-time pad encryption scheme is information-theoretically secure as long as the message is encrypted with a randomly generated one-time key which is as long as the message. However, the practicality of the one-time pad is in general ruled out given the key management issues.

Provable security is a part of modern cryptography that came to tackle the security issues where an adversary exploits vulnerabilities overlooked by the scheme designer in order to break the cryptographic scheme. It provides a rigorous mathematical framework in which schemes can be designed and their security against computationally bounded adversaries can be analysed. Essentially, in the framework of provable security we prove that a reduction exists between the difficulty of breaking (with respect to a specified adversarial model) the designed scheme and the difficulty of solving a hard problem (such as factoring large composite integers) or breaking the security of an underlying cryptographic primitive. The security of the scheme, the hard problem and possible underlying cryptographic primitives, are parameterized in terms of a security parameter and adversaries are modelled by probabilistic polynomial-time *Turing machines*. The latter means that adversaries can be seen as abstract computational devices that use randomness (for more "efficient" computation) and their running times are bounded by some polynomial in the security parameter.

As already pointed out by Bellare [15], the term provable security is *misleading* as one does not actually prove security of a scheme, but actually provides a reduction of the security of the scheme to the security of a mathematical hard problem or an underlying primitive. Hence, a more appropriate term for this genre of work would be *reductionist security*.

The first result in the direction of provable security arose soon after the earliest papers [53, 119] on public-key cryptography. In 1979 Rabin [114] introduced a signature scheme with security based on the hardness of finding square roots modulo a composite number N, and formally related the difficulty of breaking that scheme to the difficulty of factoring N. Basically, Rabin showed that the ability to extract square roots modulo N allows one to factor N in polynomial time. The drawback of Rabin's signature scheme is that it is insecure against a type of attack called *chosen-message attack* [79, 140]. Namely, the adversary could fool the signer into signing messages of its own choice and then forge a signature. It is true that this issue was noticed by Rabin and he suggested a way of overcoming this problem, but then the security reduction to factoring no longer works for the modified scheme.

Provable security emerged in the 1980s when researchers decided to develop precise definitions for cryptographic schemes and specify appropriate security models for them. In [77, 78] Goldwasser and Micali formally introduced the notion of a probabilistic public key encryption scheme, along with two strong notions of security that they called *polynomial security* and *semantic security*. Informally speaking, the former means that no polynomially bounded adversary can find two messages m_1 and m_2 whose encryptions are distinguishable. The latter means that whatever an adversary can compute about the plaintext given the corresponding ciphertext, it could also compute without the ciphertext. Goldwasser and Micali proved that every polynomially secure scheme is semantically secure and then reduced the security of their scheme, which they proved to be polynomially secure, to the problem of deciding whether a number is a quadratic residue modulo a composite integer.

The powerful idea of using reduction arguments to "prove" security of cryptographic primitives is very well known and has become standard in most cryptographic research. In the following we describe in more detail the provable security approach and discuss its limitations.

2.2.2 The Approach

The provable security paradigm is as follows:

- **Define a cryptographic scheme**. It is important to *formally* define the functionality of the scheme, specifying the behaviour of each component algorithm: Are the algorithms probabilistic or deterministc? What values are taken as inputs and what are the outputs? What are the correctness requirements?

- **Specify a security model**. A security model defines what a computationally bounded adversary is allowed to do and when, and what it means to break the scheme. The capabilities of an adversary will usually depend on a typical practical use of the cryptographic scheme and so different security models may be specified for different security goals for the same scheme. A very common approach when specifying security models is to use *game-based definitions*. Here security models can take the form of a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, where $\mathcal{C}$ answers *oracle* queries to $\mathcal{A}$, or the form of an experiment. Experiments are pseudocodes that model what inputs are given to the adversary and how they are generated. They return 0 or 1 depending on the output of the adversary. The security of a cryptographic scheme with respect to a specific security model is measured in terms of the advantage of an adversary in achieving the security goal specified by the game or experiment. An alternative approach for specifying security models is to use simulation-based approaches such as the *Universal Composability (UC) framework* [36]. Concisely, a cryptographic task is specified through an ideal functionality. A scheme is said to *securely realize* the ideal functionality if no environment (an entity who generates inputs to users, observes their outputs, and is allowed to interact with an adversary against the scheme) can distinguish the execution of the real scheme with an adversary and the execution of the ideal functionality with an ideal adversary (the simulator). The benefit of this framework is that it guarantees security of the scheme even when it is run concurrently with other schemes.

- **Provide a construction** satisfying the formal definition of the cryptographic scheme.

- **Show a reduction** from the construction to an underlying primitive or a computational hardness assumption. This step consists of showing that the only way the adversary can break the scheme, with respect to a specific security model, is by breaking the underlying primitive or the hardness assumption.

In the context of security reductions, there are a few issues that need to be taken into consideration. The first one is that there exists two approaches used to provide security reductions: the *asymptotic* approach and the *concrete* approach. Traditionally, provable security is asymptotic. It views the running time of the adversary as well as its advantage as functions of a security parameter k. The notion of *efficient algorithms* is equivalent to probabilistic algorithms running in time polynomial in k. The notion of *small advantage* is substituted by negligible advantage, meaning that the advantage is smaller than any inverse polynomial in k. A security reduction is stated by saying: *Assume assumption A holds, then scheme B is secure.* We note that this approach gives no information about the quality of the reduction.

A more practical approach to handling these issues is the concrete approach, or *practice-oriented* approach [15]. In this approach we make the parameters of the reduction explicit, making it clear how good the reduction is in terms of the advantages and running times of adversaries. Here the reduction is stated by saying: *Assume there exists an algorithm $\mathcal{B}$ which can break the security of scheme B with advantage ϵ and running time t. Then we can construct an algorithm $\mathcal{A}$ that breaks the hard problem A with advantage ϵ' and running time t'.* The relation of ϵ' and ϵ (or t' and t) may depend for example on the number of queries that the adversaries make. We say the reduction is tight if (ϵ', t') is approximately equal to (ϵ, t). A non-tight reduction means that the cryptographic scheme may need larger key sizes in order to achieve the same level of security as the underlying assumption.

The second issue concerns on which assumption one bases the security of a cryptographic scheme. There are many mathematical assumptions; some are weaker than others (for example, the factoring assumption is weaker than the RSA assumption as factoring allows one to break RSA[1]). When designing a cryptographic scheme, it is desirable to base its security on the weakest possible assumption. This is because if the assumption is shown to be wrong, then the proof is meaningless.

Succinctly, when providing security reductions, one should aim for *tight* reductions of the cryptographic scheme to as *weak* as possible assumptions. Moreover, in order not to have the security of the scheme compromised, it is important to choose

[1]Informally, the RSA assumption states that it is hard to compute e-th roots of an arbitrary integer modulo a composite N which is the product of two unknown large primes.

an appropriate security model for the practical application in which the scheme is intended to be used, which might not be an easy task. In [15], Bellare provides a detailed concrete treatment of security reductions.

We must also be aware that although very useful in providing security guarantees for complex cryptographic schemes, the provable security approach has its limitations. Usually it does not cover a class of practical attacks where an adversary can attack the physical implementation of the cryptographic scheme. This type of attack is called a *side-channel attack*. The adversary exploits leakage of information during or after the execution of the scheme. The information can be gained, for example, via measurements of power consumption [100], timing of execution [99] or electromagnetic radiation [3, 124], or even via *cold-boot attacks* [82]. The latter exploits the fact that memory contents of a computer remain readable for several seconds or minutes after power has been removed. Another type of attack considered as a side-channel attack is the case where an adversary gets side-information from error-messages coming from a decryption oracle [25, 135]. Certainly, counter-measures against side-channel attacks should be taken and so work in the direction of extending the field of provable security to overcome these attacks has already begun [4, 7, 58, 97, 110, 111].

This thesis follows the standard (meaning not capturing side-channel attacks) provable security paradigm and endeavours to attain the above desired qualities in our security reductions. We use game-based security models and our reductions are quantified in terms of the types and number of queries that adversaries against a cryptographic scheme or hard assumption can make.

2.2.3 Standard Model versus Random Oracle Model

As mentioned earlier, security models are commonly stated in terms of a game between an adversary and a challenger. A security reduction then works as follows: Let $\mathcal{B}$ be an adversary against a cryptographic scheme that can break the security notion specified in a security model, then one can construct a simulator that can break a hard problem. Usually the simulator will be the adversary, $\mathcal{A}$, against the hard problem and it must be able to simulate $\mathcal{B}$'s challenger behaviour consistently.

This means that $\mathcal{A}$ must be able to answer $\mathcal{B}$'s queries correctly. Such a reduction, without any additional assumptions, is said to be a proof in the *standard model*.

Designing cryptographic schemes which are secure in the standard model is not easy. Particularly, what makes it difficult to achieve standard model security is the problem of answering the adversary's queries correctly. In order to overcome this difficulty and to simplify the analyses of cryptographic schemes, Bellare and Rogoway [18] introduced the *random oracle model* (ROM).[2] This model assumes the existence of a public *random oracle* H, which is an oracle function, or a "black-box", that responds to queries with values chosen uniformly at random from its output domain, except that for any specific query the oracle always responds with the same value every time it receives that query. For an input x, the only way to get $\mathsf{H}(x)$ is to explicitly ask the oracle. Furthermore, the inputs x to the oracle are private, meaning that no adversary can see the inputs x queried by a user.

In practice, the random oracles assumed in security proofs of cryptographic schemes are instantiated with hash functions. This seems intuitive since it would be desirable that the outputs of a well-designed hash function appear to be completely indepent even for related inputs.

The random oracle methodology has been widely employed to prove security of efficient cryptographic schemes. However, the problem with this methodology is that, as shown by Canetti, Goldreich and Halevi [38, 39], a scheme proven secure in the ROM is not necessarily secure when the random oracle is instantiated with any hash function. But then the question is: why are people still using the ROM methodology? The point is that proving security in the ROM is better than having no security proof at all and if there is an attack exploiting some specific properties of the chosen hash function, the same scheme can still be used by instantiating the random oracle with a different hash function. Moreover, there is a class of attacks called *generic attacks* where the adversary does not exploit any particular structural property of the hash function. Arguably, in this situation the ROM proofs apply and do bring some security guarantees. In addition, the best schemes provably secure in the ROM tend to be more efficient than the best standard model secure schemes.

[2]The random oracle model was earlier used, but without naming it so, by Fiat and Shamir [59].

Despite some practicality offered by random oracles, they do not offer security in general and so proofs in the standard model (i.e. without random oracles) are preferable to proofs in the ROM. Finally, it is important to stress here that not all schemes that use cryptographic hash functions require random oracles. There are schemes that require only some properties of the hash functions, such as *collision resistance* (see Section 2.3.1), and that can often be proven secure in the standard model. This will be the case in some of the constructions provided in this thesis.

2.2.4 Proof Techniques

A useful method for proving security of cryptographic schemes is to use game hopping proofs [51, 132]. A game hopping proof is structured as a sequence of games, instead of a single game. This approach has been extensively used to prove security of complex cryptographic schemes. In many circumstances it makes proofs clearer and less error-prone. In this thesis we also make use of this very useful technique in most of our proofs, and hence we recall it here.

Typically, in a game hopping proof we consider a sequence of games $G_0, G_1, \ldots, G_\lambda$ all defined over the same underlying probability space. Depending on the type of security goal we want to achieve, *computational security* or *indistinguishability*, the games will take different shape. Let us assume for now we want to prove an indistinguishability-based security notion. Commonly, in this case a challenger uniformly chooses a bit $b \in \{0, 1\}$ which the adversary has to guess. We start with G_0, the actual adversarial game. Then we make successive transitions, in such a way that the adversary's view is indistinguishable among successive games, until we reach game G_λ for which the adversary's probability of success in outputting the correct bit b is clearly $1/2$. Here the adversary's advantage is the absolute value of the difference between its success probability in game G_0 and $1/2$. Note that, as we will see in security proofs provided in this thesis, this is not the only way to define a sequence of games.

In the case of a computational-based security goal, G_0 is simply the original adversarial game described in the security model. We make slight modifications to successive games, starting from G_0, until we reach game G_λ, where the adversary's

success probability can easily be computed.

The transitions between successive games are frequently limited to the following types. Let $\Pr[S_\iota]$ be the success probability of the adversary in game G_ι.

- **Transitions based on indistinguishability**. The idea here is that if an adversary can identify the small change between two successive games, then it is possible to construct an adversary that tells apart two distributions that were assumed to be indistinguishable.
- **Transitions based on small failure events**. Two successive games $G_{\iota-1}, G_\iota$ are identical unless a certain "failure" event $\mathtt{F}$ occurs. This leads to the *Difference Lemma* [132]: "Let $S_{\iota-1}, S_\iota$ be the events that an adversary is successful in games $G_{\iota-1}, G_\iota$, respectively, and let $\mathtt{F}$ be a failure event such that $S_{\iota-1} \wedge \neg\mathtt{F} \Leftrightarrow S_\iota \wedge \neg\mathtt{F}$. Then $|\Pr[S_{\iota-1}] - \Pr[S_\iota]| \leq \Pr[\mathtt{F}]$." So if $\Pr[\mathtt{F}]$ is small (i.e. negligible), the two games $G_{\iota-1}$ and G_ι are computationally indistinguishable.
- **Bridging steps**. These steps are used to help the next transition so that the proof is easier to follow. The change made is entirely conceptual and so the adversary's success probability remains the same.
- **Transitions based on large failure events**. This type of transition is similar to the one based on small failure events, but here it is assumed that the probability that a failure event $\mathtt{F}$ occurs is large (but not overwhelming) and that $\mathtt{F}$ is independent of $S_{\iota-1}$. Is is also assumed that $\neg\mathtt{F}$ is non-negligible. The two types of security goal, computational security and indistinguishability, are considered separately. For the former case the goal is to directly relate $\Pr[S_{\iota-1}]$ and $\Pr[S_\iota]$, in which case the relation $\Pr[S_\iota] = \Pr[\neg\mathtt{F}]\Pr[S_{\iota-1}]$ can be obtained. For the latter case, the goal is to show that $|\Pr[S_\iota] - 1/2|$ is negligible if and only if $|\Pr[S_{\iota-1}] - 1/2|$ is negligible, which can be demonstrated by showing that $|\Pr[S_\iota] - 1/2| = \Pr[\neg\mathtt{F}]|\Pr[S_{\iota-1}] - 1/2|$.

The first three possible types of transitions were identified by Shoup [132] and the fourth by Dent [51].

Hybrid arguments. A special type of game-hopping proof is when the proof consists of a sequence of transitions based on *indistinguishability*. It is commonly called

a hybrid argument, as the computational indistinguishability of the two extreme games is proven based on the indistinguishability property of successive hybrids (modifications of the original games). The distinguishability gap of the extreme games (also called extreme hybrids) is inversely proportional to the total number of games. Thus, it is essential that the number of hybrids is small (i.e. polynomial). We will see an example of a hybrid argument in the proof of Theorem 3.1.

2.3 Cryptographic Primitives

We now review and formally define the cryptographic primitives used in the schemes presented in this thesis.

2.3.1 Hash Functions

In general, hash functions are polynomial-time functions that take as input strings of arbitrary length and compress them into shorter strings. Formally, we will deal with a family of hash functions indexed by a key. So the hash function will be a two-input function that takes as input a key in the key domain and an arbitrary-length string, and outputs a hash value. (In practice, hash functions have a maximum input size and they are non-keyed. The concept of keyed hash functions was introduced to help the formalization of security properties of hash functions [49].) There are many different types of hash functions with different security properties. Here we recall the ones that we will consider in this thesis.

Collision Resistant Hash Functions. Let $\mathsf{CRF} : \mathcal{F} \times \mathcal{M} \to \mathcal{Y}$ be a family of keyed hash functions and let $\mathcal{A}_{\mathcal{H}}$ be an adversary. (For each key $f \in \mathcal{F}$, there is a hash function $\mathsf{CRF}_f : \mathcal{M} \to \mathcal{Y}$.) Here $\mathcal{M}$ is the domain, $\mathcal{F}$ is the key space and $\mathcal{Y}$ is the range of CRF. For $m \in \mathcal{M}$ and $f \in \mathcal{F}$ we write $\mathsf{CRF}_f(m) = \mathsf{CRF}(f, m)$. CRF is said to be *collision resistant* if, for a hash function $\mathsf{CRF}_f \in \mathsf{CRF}$ (where the hash key f is chosen at random from $\mathcal{F}$), it is infeasible for any adversary $\mathcal{A}_{\mathcal{H}}$ to find two distinct values m and m' such that $\mathsf{CRF}_f(m) = \mathsf{CRF}_f(m')$. More formally, following [120], we define

$$\mathrm{Adv}^{\mathrm{coll}}_{\mathcal{A}_{\mathcal{H}},\mathsf{CRF}}(k) = |\Pr[f \leftarrow \mathcal{F}; (m, m') \leftarrow \mathcal{A}_{\mathcal{H}}(f) : (m \neq m') \wedge (\mathsf{CRF}_f(m) = \mathsf{CRF}_f(m'))]|.$$

The hash function family is said to be *collision resistant* if $\mathrm{Adv}^{\mathrm{coll}}_{\mathcal{A}_{\mathcal{H}},\mathsf{CRF}}(k)$ is negligible for any polynomial-time adversary $\mathcal{A}_{\mathcal{H}}$.

Target Collision Resistant Hash Functions. The difference between a target collision resistant hash function TCR_f and a collision resistant hash function CRF_f is that, in the former case, it is infeasible for any adversary, given a value m, to find a distinct value m' such that $\mathsf{TCR}_f(m) = \mathsf{TCR}_f(m')$. More formally, we define

$$\begin{aligned}\mathrm{Adv}^{\mathrm{TCR}}_{\mathcal{A}_{\mathcal{H}},\mathsf{TCR}}(k) \;=\; \big|\Pr\big[f \leftarrow \mathcal{F}, m \leftarrow \mathcal{M};\, m' \leftarrow \mathcal{A}_{\mathcal{H}}(f,m) :\\ (m \neq m') \wedge (\mathsf{TCR}_f(m) = \mathsf{TCR}_f(m'))\big]\big|\,.\end{aligned}$$

The hash function family TCR is said to be *target collision resistant* if $\mathrm{Adv}^{\mathrm{TCR}}_{\mathcal{A}_{\mathcal{H}},\mathsf{TCR}}(k)$ is negligible for any polynomial-time adversary $\mathcal{A}_{\mathcal{H}}$.

Chameleon Hash Functions. Chameleon hash functions [102] can be thought of as collision resistant hash functions with a trapdoor for finding collisions. Let k be a security parameter. A chameleon hash function $\mathsf{ChamH} : \mathcal{D} \times \mathcal{R}_{\mathrm{Cham}} \to \mathcal{I}$, where $\mathcal{D}$ is the domain, $\mathcal{R}_{\mathrm{Cham}}$ the randomness space and $\mathcal{I}$ the range, is associated with a pair of public and private keys (the latter called a trapdoor). These keys are denoted respectively by hk and ck and are generated by a PPT algorithm $\mathsf{Cham.KeyGen}(1^k)$. The public key hk defines a chameleon hash function, denoted $\mathsf{ChamH}_{hk}(\cdot,\cdot)$. On input a message $m \in \mathcal{D}$ and a random string $r \in \mathcal{R}_{\mathrm{Cham}}$, this function generates a hash value $\mathsf{ChamH}_{hk}(m,r)$ which satisfies the following properties:

Collision resistance. There is no efficient algorithm that on input the public key hk can find pairs (m_1, r_1) and (m_2, r_2) where $m_1 \neq m_2$ such that we have $\mathsf{ChamH}_{hk}(m_1, r_1) = \mathsf{ChamH}_{hk}(m_2, r_2)$, except with some probability that is negligible in k.

Trapdoor collisions. There is an efficient algorithm $\mathsf{Cham.Trap}$ that on input the private key ck, any pair (m_1, r_1) and any additional message m_2, finds a value r_2 such that $\mathsf{ChamH}_{hk}(m_1, r_1) = \mathsf{ChamH}_{hk}(m_2, r_2)$. Also, for uniformly and independently chosen m_1, r_1 and m_2, the value of r_2 is independently and uniformly distributed over $\mathcal{R}_{\mathrm{Cham}}$.

Uniformity. All mesages m induce the same probability distribution on the output of $\mathsf{ChamH}_{hk}(m,r)$ for r chosen uniformly at random. This property prevents a

third party, examining the hash value, from deducing any information about the hashed message.

More formally, we define the advantage of an adversary $\mathcal{A}_{\mathcal{CH}}$ against ChamH as

$$\begin{aligned}\mathrm{Adv}^{\mathrm{coll}}_{\mathcal{A}_{\mathcal{CH}},\mathsf{ChamH}}(k) \;=\; \Big|\Pr\Big[&hk \leftarrow \mathsf{Cham.KeyGen}(1^k); (m_1, r_1, m_2, r_2) \leftarrow \mathcal{A}_{\mathcal{CH}}(hk) : \\ &(m_1 \neq m_2) \wedge (\mathsf{ChamH}_{hk}(m_1, r_1) = \mathsf{ChamH}_{hk}(m_2, r_2))\Big]\Big| \,.\end{aligned}$$

The composition of a chameleon hash function and a (regular) collision resistant hash function (where the latter is applied first) results in a chameleon hash function. In [102], there are chameleon hash function constructions based on standard assumptions like the DL assumption and the factoring assumption.

2.3.2 Programmable Hash Functions

Programmable hash functions (PHFs) were proposed by Hofheinz and Kiltz [86] as an abstraction of random oracles that can also be instantiated in the standard model. In a nutshell, a PHF H is a keyed group hash function that maps a bitstring X (e.g. a message to be signed) to a group element $\mathsf{H}(X)$. The fundamental property of H is that it can behave in two indistinguishable ways. If the *standard key generation algorithm* is used, a key κ is produced and the hash function normally hashes its inputs into group elements; however, if an alternative *trapdoor key generation algorithm* is used, then a key κ' (which is indistinguishable from κ) and a trapdoor t are generated. This special trapdoor allows $\mathsf{H}(X)$ to be decomposed in the form $c^{\alpha_X} h^{\beta_X}$ for previously chosen c, h. In a larger proof, c will usually be a "challenge element" (e.g. a part of a given Diffie-Hellman challenge), so that $\mathsf{H}(X)$ contains a challenge component if and only if $\alpha_X \neq 0$.

PHFs can be used to employ partitioning strategies: during a security proof a simulator can, with some non-negligible probability, partition the set of all PHF inputs into two disjoint sets: the set of inputs for which it hopes to embed the challenge and the set of inputs for which it hopes to be able to consistently answer adversarial queries. For example, Waters' CDH-based signature scheme [138] (implicitly) uses a PHF to partition the set of all messages into "signable" and "unsignable" messages. (In his case, a message X is signable if and only if $\alpha_X \neq 0$.) During the proof of

unforgeability, we hope that all messages for which an adversary requests a signature are signable, while the adversary's forgery corresponds to an unsignable message.

Let us now recall the formal definition of PHFs from [86]. Let k be a security parameter, $\ell(\cdot)$ a polynomial and GPgen a group generator. A group hash function $\mathsf{H} : \{0,1\}^{\ell(k)} \to \mathbb{G}$, with $\mathbb{G} \leftarrow \mathsf{GPgen}(1^k)$, consists of two polynomial-time algorithms: $\mathsf{PHF.Gen}$ and $\mathsf{PHF.Eval}$. The key generation algorithm $\mathsf{PHF.Gen}$ is probabilistic and, on input the security parameter, generates a hash key κ. The evaluation algorithm $\mathsf{PHF.Eval}$ is deterministic and uses κ to evaluate H on input a string $X \in \{0,1\}^{\ell(k)}$; we write $\mathsf{H}_\kappa(X) := \mathsf{PHF.Eval}(\kappa, X) \in \mathbb{G}$.

Definition 2.26 (PHF)**.** *A group hash function* $\mathsf{H} = (\mathsf{PHF.Gen}, \mathsf{PHF.Eval})$ *is said to be* (m, n, γ, δ)*-programmable if there is a probabilistic polynomial-time* trapdoor key generation algorithm $\mathsf{PHF.TrapGen}$ *and a deterministic polynomial-time* trapdoor evaluation algorithm $\mathsf{PHF.TrapEval}$ *with the following properties:*

Syntatics. *For group elements* $c, h \in \mathbb{G}$, $(\kappa', t) \leftarrow \mathsf{PHF.TrapGen}(1^k, c, h)$. *Moreover, on input* $X \in \{0,1\}^{\ell(k)}$, $(\alpha_X, \beta_X) \leftarrow \mathsf{PHF.TrapEval}(t, X)$ *with* $\alpha_X, \beta_X \in \mathbb{Z}$.

Correctness. *For all generators* $c, h \in \mathbb{G}$ *and all possible pair of hash key and trapdoor information* (κ', t) *output by* $\mathsf{PHF.TrapGen}(1^k, c, h)$*, and for all* $X \in \{0,1\}^{\ell(k)}$ *and the corresponding pairs of integers* (α_X, β_X) *output by* $\mathsf{PHF.TrapEval}(t, X)$*, the evaluation algorithm satisfies*

$$\mathsf{H}_{\kappa'}(X) = \mathsf{PHF.Eval}(\kappa', X) = c^{\alpha_X} h^{\beta_X}.$$

Statistically close trapdoor keys. *For all generators* $c, h \in \mathbb{G}$*, the keys* $\kappa \leftarrow \mathsf{PHF.Gen}(1^k)$ *and* $\kappa' \leftarrow \mathsf{PHF.TrapGen}(1^k, c, h)$ *are statistically* γ*–close:* $\kappa \overset{\gamma}{\equiv} \kappa'$.

Well-distributed logarithms. *For all generators* $c, h \in \mathbb{G}$ *and all possible* (κ', t) *output by* $\mathsf{PHF.TrapGen}(1^k, c, h)$*, for all* $X_1, \ldots, X_m, Z_1, \ldots, Z_n \in \{0,1\}^{\ell(k)}$ *with* $X_i \neq Z_j$*, and for the corresponding integers* $(\alpha_{X_i}, \beta_{X_i}) \leftarrow \mathsf{PHF.TrapEval}(t, X_i)$ *and* $(\alpha_{Z_j}, \beta_{Z_j}) \leftarrow \mathsf{PHF.TrapEval}(t, Z_j)$*, we have*

$$\Pr\left[\alpha_{X_1} = \ldots = \alpha_{X_m} = 0 \wedge \alpha_{Z_1}, \ldots, \alpha_{Z_n} \neq 0\right] \geq \delta,$$

where the probability is over the trapdoor t *that was produced along with* κ'.

If γ is negligible and δ is noticeable, H is simply called (m, n)-programmable. Furthermore, H is said to be (m, poly)-programmable if it is (m, q)-programmable for every polynomial $q = q(k)$. Respectively, H is (poly, n)-programmable if the analogous condition holds.

Note that, in the above definition, c and h need not be generators and $\mathbb{G}$ need not be cyclic. However, we also want to be able to substitute PHFs with random oracles, as we will see in Section 4.5.2. In such an analysis, the images of H should appear uniformly and independently distributed to an adversary who sees only public information. This is then guaranteed if $\mathbb{G}$ is cyclic and c, h are generators.

A useful variant of the standard definition of a PHF is the concept of a *weak* (m, n, γ, δ)-PHF [89], which is essentially an (m, n, γ, δ)-PHF according to Definition 2.26, except that in the former case the trapdoor key generation algorithm $\mathsf{PHF.TrapGen}$ receives as additional input a list of strings $X_1, \ldots, X_m \in \{0,1\}^{\ell(k)}$, and the well-distributed logarithms property holds only with respect to this list of X_i ($i \in [m]$) and any set of strings $Z_1, \ldots, Z_n \in \{0,1\}^{\ell(k)}$. Weak programmable hash functions are usually more efficient than standard PHFs and are very useful to prove cryptographic schemes to be weakly secure (i.e. when the adversary has to commit to a list of values to be queried before seeing the public key). Full security (i.e. when the adversary is fully adaptive) can often be achieved by using weak PHFs together with chameleon hashes.

Definition 2.27 (Weak PHF)**.** *A group hash function H is called a* weak (m, n, γ, δ)*-programmable hash function if it is a PHF in the sense of Definition 2.26, but with the following differences:*

- *For group elements $c, h \in \mathbb{G}$ and set of strings $X_1, \ldots, X_m \in \{0,1\}^{\ell(k)}$, $(\kappa', t) \leftarrow \mathsf{PHF.TrapGen}(1^k, c, h, X_1, \ldots, X_m)$.*
- *The well-distributed logarithms property from Definition 2.26 holds only for the given set of strings $X_1, \ldots, X_m \in \{0,1\}^{\ell(k)}$ (and any $Z_1, \ldots, Z_n \in \{0,1\}^{\ell(k)}$, with $Z_j \neq X_i$).*

Example of a weak $(2, \mathbf{poly})$-PHF. [3] Let $\mathbb{G} \leftarrow \mathsf{GPgen}(1^k)$ be a group of prime order p. Define $\mathsf{H}_{\mathsf{weak}} = (\mathsf{PHF.Gen}, \mathsf{PHF.Eval})$ as follows.

- $\mathsf{PHF.Gen}(1^k)$ returns $\kappa = (u_0, u_1, u_2) \leftarrow \mathbb{G}$.
- On input $X \in \{0,1\}^{\ell(k)}$, $\mathsf{PHF.Eval}(\kappa, X)$ returns $\mathsf{PHF.Eval}(\kappa, X) = u_0 u_1^X u_2^{X^2}$. (Here the $\ell(k)$-bit string X is interpreted as an integer.)

To see that $\mathsf{H}_{\mathsf{weak}}$ is a weak $(2, \text{poly})$-PHF consider the following algorithms:

- $\mathsf{PHF.TrapGen}(1^k, c, h, X_1, X_2)$ samples random integers $q_0, q_1, q_2 \leftarrow \mathbb{Z}_p$ and then computes the coefficients $p_0, p_1, p_2 \in \mathbb{Z}_p$ of the polynomial

$$p(t) = p_0 + p_1 t + p_2 t^2 = (t - X_1)(t - X_2).$$

  It then sets $u_i' = c^{p_i} h^{q_i}$ $(i = 0, 1, 2)$ and outputs $\kappa' = (u_0', u_1', u_2')$ and $t = (p_0, p_1, p_2, q_0, q_1, q_2)$.

- $\mathsf{PHF.TrapEval}(t, X)$ outputs $(\alpha_X = p(X), \beta_X = q(X))$, where $q(X) = q_0 + q_1 X + q_2 X^2$.

Syntactics and correctness are obviously satisfied. Now note that since $q_i \leftarrow \mathbb{Z}_p$, we have $u_i \leftarrow \mathbb{G}$. This implies $\gamma = 0$. Furthermore from the construction of $\mathsf{PHF.TrapGen}$ we see that $\alpha_X = 0$ if and only if $X \in \{X_1, X_2\}$ and therefore $m = 2$, $n = \text{poly}(k)$ and $\delta = 1$.

2.3.3 Pseudorandom Functions

Let $\mathsf{F} : \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a function family. Here $\mathcal{K}$ is the set of keys, $\mathcal{D}$ the domain, and $\mathcal{R}$ the range of F. For all $\kappa \in \mathcal{K}$ and $x \in \mathcal{D}$ we also write $\mathsf{F}_\kappa(x) = \mathsf{F}(\kappa, x)$ and call $\mathsf{F}_\kappa : \mathcal{D} \to \mathcal{R}$ an *instance* of F. Informally, a function family F is said to be a pseudorandom function (PRF) [75, 76] if the input-output behaviour of a random instance of the family, F_κ, is computationally indistinguishable from that of a truly random function with the same domain and range. This means that it is infeasible

[3] This is the same weak-PHF (for $m = 2$) as the one given in [89, Definition 8]. There the authors prove $\mathsf{H}_{\mathsf{weak}}$ to be a weak $(m, 1)$-PHF. We stress that a PHF can be programmed with different parameters.

for anyone with only black-box access to a function (where no information is gained about the function, other than its output value for any arbitrarily chosen argument) to tell F_κ and a truly random function apart.

The term "truly random function" means that the function is chosen at random from the set of all functions mapping $\mathcal{D}$ to $\mathcal{R}$, which we denote $\mathsf{Rand}(\mathcal{D},\mathcal{R})$. Note that if we let $\mathcal{D} = \{0,1\}^\ell$ and $\mathcal{R} = \{0,1\}^L$, the cardinality of $\mathsf{Rand}(\mathcal{D},\mathcal{R})$ is 2^{L2^ℓ}. Thus, we would need $L2^\ell$ bits to specify a function in $\mathsf{Rand}(\mathcal{D},\mathcal{R})$, which is impractical. A more intuitive way to think of a black-box access to a random function $f_0 : \mathcal{D} \to \mathcal{R}$ is to think that each time the box is given an input $x \in \mathcal{D}$, it returns a random element of $\mathcal{R}$ with the constraint that it always returns the same value for the same input x.

We now provide a formal definition of a PRF.

Definition 2.28 (PRF)**.** *Let $\mathsf{F} : \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a function family and let $\mathcal{A}_\mathsf{F}$ be an algorithm that has oracle access to a function $f_\beta : \mathcal{D} \to \mathcal{R}$ as defined in the following experiment.*[4]

$$\begin{array}{l}\text{Experiment } \mathrm{Exp}^{\mathrm{PRF},\beta}_{\mathcal{A}_\mathsf{F},\mathsf{F}}(1^k)\\ \quad f_0 \leftarrow \mathsf{Rand}(\mathcal{D},\mathcal{R}); \kappa \leftarrow \mathcal{K}, f_1 \leftarrow \mathsf{F}_\kappa\\ \quad \beta' \leftarrow \mathcal{A}^{f_\beta}_\mathsf{F}(1^k)\\ \quad \textit{Return } \beta'\end{array}$$

The advantage of $\mathcal{A}_\mathsf{F}$ in the above experiment is defined as

$$\mathrm{Adv}^{\mathrm{PRF}}_{\mathcal{A}_\mathsf{F},\mathsf{F}}(k) = \left|\Pr\left[\mathrm{Exp}^{\mathrm{PRF},1}_{\mathcal{A}_\mathsf{F},\mathsf{F}}(1^k) = 1\right] - \Pr\left[\mathrm{Exp}^{\mathrm{PRF},0}_{\mathcal{A}_\mathsf{F},\mathsf{F}}(1^k) = 1\right]\right|,$$

where the probability is taken over the random choices made in the experiment.

We say that F is a PRF if $\mathrm{Adv}^{\mathrm{PRF}}_{\mathcal{A}_\mathsf{F},\mathsf{F}}(k)$ is negligible for every polynomial-time adversary $\mathcal{A}_\mathsf{F}$.

It should be noted that the function family F is public; anyone who knows a key $\kappa \in \mathcal{K}$ can compute $\mathsf{F}_\kappa(x)$ for some $x \in \mathcal{D}$ – obviously the key κ in the above experiment is not known to the adversary $\mathcal{A}_\mathsf{F}$. Furthermore, F is a family of efficiently computable functions.

[4] More precisely, we assume that $\mathcal{K}, \mathcal{D}, \mathcal{R}$ are families of finite sets, indexed by a security parameter k. That is, we require $\mathcal{K} = (\mathcal{K}_k)_{k\in\mathbb{N}}$, and similarly for $\mathcal{D}$ and $\mathcal{R}$. For the sake of a cleaner exposition, however, we do not write down the security parameter explicitly.

2.3.4 Pseudorandom Generators

A *pseudorandom generator* (PRG) is a deterministic algorithm that stretches a truly random binary sequence, called the *seed*, into a longer (polynomially-bounded) binary sequence that "looks" random, called the *pseudorandom sequence*.[5] We formalize this notion in the following definition.

Definition 2.29 (PRG)**.** *Let* $\mathsf{G} : \{0,1\}^k \to \{0,1\}^{p(k)}$ *be a deterministic polynomial-time function and let* $\mathcal{D}$ *be an algorithm. Consider the following experiment.*

$$\begin{array}{l}\text{Experiment } \mathrm{Exp}^{\mathrm{PRG},\beta}_{\mathcal{D},\mathsf{G}}(1^k)\\ \quad T_0 \leftarrow \{0,1\}^{p(k)}; x \leftarrow \{0,1\}^k, T_1 \leftarrow \mathsf{G}(x)\\ \quad \beta' \leftarrow \mathcal{D}(T_\beta)\\ \quad \textit{Return } \beta'\end{array}$$

We define $\mathcal{D}$*'s advantage in the above experiment as*

$$\mathrm{Adv}^{\mathrm{PRG}}_{\mathcal{D},\mathsf{G}}(k) = \left|\Pr\left[\mathrm{Exp}^{\mathrm{PRG},1}_{\mathcal{D},\mathsf{G}}(1^k) = 1\right] - \Pr\left[\mathrm{Exp}^{\mathrm{PRG},0}_{\mathcal{D},\mathsf{G}}(1^k) = 1\right]\right|,$$

where the probability is taken over all choices of x *and the random choices made by the distinguisher* $\mathcal{D}$*. We say* G *is a pseudorandom generator (PRG) if* $p(k) > k$ *and* $\mathrm{Adv}^{\mathrm{PRG}}_{\mathcal{D},\mathsf{G}}(k)$ *is negligible for every polynomial-time distinguisher* $\mathcal{D}$*. Then for random* $x \in \{0,1\}^k$*,* $\mathsf{G}(x) \in \{0,1\}^{p(k)}$ *is said to be a pseudorandom sequence. By definition,* $\mathsf{G}(x)$ *is computationally indistinguishable from a random bit sequence of length* $p(k)$*.*

We now define a particular pseudorandom generator, introduced by Blum, Blum and Shub [26].

Definition 2.30 (The BBS pseudorandom generator)**.** *Let* N *be a Blum integer, i.e.* $N = PQ$ *where* P*,* Q *are distinct primes with* $P = Q = 3 \pmod 4$*. Let* x *be a quadratic residue modulo* N*, i.e.* $x \in \mathbb{QR}_N$*. (For consistency with Definition 2.29, we assume that elements in* $\mathbb{QR}_N$ *can be viewed as bit strings of length* k*.) We establish the following notation:* $\mathsf{LSB}_N(x) = (x \pmod N) \pmod 2$ *(the least significant bit of* $x \pmod N$*). The BBS pseudorandom generator* BBS *applied to* x *and modulus* N *is defined to have output:*

$$\mathsf{BBS}_N(x) = (\mathsf{LSB}_N(x), \mathsf{LSB}_N(x^2), \ldots, \mathsf{LSB}_N(x^{2^{\ell-1}})) \in \{0,1\}^\ell,$$

[5] We remark that PRGs are often called PRNGs (pseudorandom number generators) or PRBGs (pseudorandom bit generators).

where $\ell = \ell(k) > k$.

The result below, proved in [88] and based on results from [6, 26, 28] states that any BBS-distinguisher can be used to factor Blum integers.

Theorem 2.8 (BBS-distinguisher ⇒ factoring algorithm)**.** *Let* RSAgen *be a probabilistic polynomial-time algorithm that on input a security parameter* 1^k *generates a Blum integer* N*. Let* BBS *be the BBS generator as in Definition 2.30. For every PPT algorithm* $\mathcal{D}$ *that succeeds in breaking* BBS *with advantage* $\mathrm{Adv}^{\mathrm{PRG}}_{\mathcal{D},\mathsf{BBS}}(k)$ *running in time* t_{BBS}*, there exists a PPT algorithm* $\mathcal{A}$ *that factors* N *with advantage* $\mathrm{Adv}^{\mathrm{fac}}_{\mathcal{A},\mathsf{RSAgen}}(k)/\ell$*, where* ℓ *is the size of the BBS generator's output. Algorithm* $\mathcal{A}$ *runs in time* $t_{fac} \approx k^4 t_{BBS}/(\mathrm{Adv}^{\mathrm{PRG}}_{\mathcal{D},\mathsf{BBS}}(k))^2$*. This reduction holds even if the modulus* N *and the* 2^ℓ*-th power* x^{2^ℓ} *of the BBS seed* x *are published.*

We note that the security of the BBS generator still holds if it simultaneously outputs up to $\log_2 \log_2 N$ least significant bits of each $x^{2^i} \pmod N$ instead of only the least significant bit [6, 136]. The downside of this approach is that the tightness of the concrete security reduction in [6, 136] is compromised.

We stress that Theorem 2.8 is valid not only when inputs for the BBS generator are taken from $\mathbb{QR}_N$ as shown in [88], but also when the inputs are taken from the group $\mathbb{QR}_N^+$ as shown in [90] (the journal version of [88]).

Forward-secure PRG. A stronger notion of security for PRGs is the notion of forward security. Forward-secure PRGs (FS-PRGs), introduced by Bellare and Yee in [19], are stateful/iterated PRGs that deterministically derive sequences of fixed-length bit strings from an initial (random) seed. More precisely, in each iteration they output a string of bits, update their internal state, and securely erase the old state. Like in regular PRGs, the output sequences are required to be indistinguishable from sequences of random strings. The pivotal property of FS-PRGs is *forward security*, i.e. the adversary has the ability to eventually corrupt the generator's internal state, but indistinguishability of output strings is guaranteed to still hold up to that point. Different constructions of FS-PRGs were proposed in [19], including highly efficient ones based on symmetric building blocks like block ciphers or HMAC [20], and also a construction based on a number-theoretic assumption (specifically, on the Blum-Blum-Shub PRG [26]).

We recall here the definition and security notion of FS-PRGs. Observe that we slightly weaken the model from [19] (considering static adversaries instead of adaptive ones). The FS-PRG constructions proposed and proved secure in [19] naturally remain secure in our adapted model.

Definition 2.31 (FS-PRG)**.** *Let* $\mathsf{G_{FS}} = (\mathsf{G_{FS}.Setup}, \mathsf{G_{FS}.Key}, \mathsf{G_{FS}.Next})$ *be a set of efficient algorithms such that* $\mathsf{G_{FS}.Setup}$ *is a probabilistic algorithm that, on input a security parameter* 1^k*, outputs a set of system parameters 'params';* $\mathsf{G_{FS}.Key}$ *is a probabilistic key generation algorithm that takes 'params' as input and outputs an initial state* $St_0 \in \{0,1\}^k$ *(the initial seed); and* $\mathsf{G_{FS}.Next} : \{0,1\}^k \to \{0,1\}^k \times \{0,1\}^{p(k)}$ *is a deterministic algorithm that turns state* $St_{i-1} \in \{0,1\}^k$ *(the 'seed' at iteration* i*) into a pair* (St_i, Out_i)*, where* $St_i \in \{0,1\}^k$ *is the updated state, and* Out_i *is a* $p(k)$*-bit string.*

Let $\mathcal{D}$ *be an adversary against* $\mathsf{G_{FS}}$*. The adversary* $\mathcal{D}$ *is fed with a number of output blocks,* $Out_1, Out_2, \ldots, Out_i$*, each of length* $p(k)$*, and is given the then current state of the generator,* St_i*. Its job is to decide whether these blocks are the real output of the generator, or just a sequence of random bits. To formalize this, we consider the following experiment.*

$$
\begin{array}{l}
\text{Experiment } \mathrm{Exp}^{\text{FS-PRG},\beta}_{\mathcal{D},\mathsf{G_{FS}}}(1^k) \\
\quad i \leftarrow \mathcal{D} \\
\quad params \leftarrow \mathsf{G_{FS}.Setup}(1^k) \\
\quad St_0 \leftarrow \mathsf{G_{FS}.Key}(params) \\
\quad i' \leftarrow 0 \\
\quad \text{Repeat} \\
\quad\quad i' \leftarrow i' + 1 \\
\quad\quad (St_{i'}, Out_{i'}) \leftarrow \mathsf{G_{FS}.Next}(St_{i'-1}) \\
\quad\quad \text{If } \beta = 0,\ Out_{i'} \leftarrow \{0,1\}^{p(k)} \\
\quad \text{Until } i' = i \\
\quad Out \leftarrow Out_1, Out_2, \ldots, Out_i \\
\quad \beta' \leftarrow \mathcal{D}(St_i, Out) \\
\quad \textit{Return } \beta'
\end{array}
$$

The advantage of $\mathcal{D}$ *is defined as*

$$\mathrm{Adv}^{\text{FS-PRG}}_{\mathcal{D},\mathsf{G_{FS}}}(k) = \left|\Pr\left[\mathrm{Exp}^{\text{FS-PRG},1}_{\mathcal{D},\mathsf{G_{FS}}}(1^k) = 1\right] - \Pr\left[\mathrm{Exp}^{\text{FS-PRG},0}_{\mathcal{D},\mathsf{G_{FS}}}(1^k) = 1\right]\right| \ .$$

We say that $\mathsf{G_{FS}}$ *is an FS-PRG if* $\mathrm{Adv}^{\text{FS-PRG}}_{\mathcal{D},\mathsf{G_{FS}}}(k)$ *is negligible for every polynomial-time adversary* $\mathcal{D}$*.*

The BBS generator as an FS-PRG. In [19], Bellare and Yee show that number-theoretic PRGs like the BBS can be modified to be seen as forward secure – in accordance with our FS-PRG definition, $\mathsf{G_{FS}.Setup}(1^k)$ would generate a Blum integer N; the random seed $x_0 \in \mathbb{QR}_N$ (initial state) would be generated by $\mathsf{G_{FS}.Key}(N)$; and $\mathsf{G_{FS}.Next}$ on input a state $St_{i-1} = x$ would output the next state $St_i = x^2 \pmod N$ and $Out_i = \mathsf{LSB}_N(x)$. If ℓ output blocks are produced, this construction gives us the standard BBS PRG from Definition 2.30.[6] We remark that in order to output larger output blocks, one can adapt this construction in the following way:

Construction 2.1. *The algorithms of a forward secure BBS pseudorandom generator* $\mathsf{BBS} = (\mathsf{G_{FS}.Setup}, \mathsf{G_{FS}.Key}, \mathsf{G_{FS}.Next})$ *which outputs* ℓ *bits per block are as follows:* $\mathsf{G_{FS}.Setup}(1^k)$ *generates a Blum integer* N*;* $\mathsf{G_{FS}.Key}(N)$ *outputs a seed* $x_0 \in \mathbb{QR}_N$*; on input a state* $St_{i-1} = x$*, algorithm* $\mathsf{G_{FS}.Next}(x)$ *outputs the next state* $St_i = x^{2^\ell} \pmod N$ *and an* ℓ*–bit block* $Out_i = \mathsf{BBS}_N(x) = (\mathsf{LSB}_N(x), \mathsf{LSB}_N(x^2), \ldots, \mathsf{LSB}_N(x^{2^{\ell-1}}))$.

From Theorem 2.8 it is easy to see that Construction 2.1 is actually an FS-PRG if factoring Blum integers is hard.

2.3.5 Key Encapsulation Mechanism

A *key encapsulation mechanism* (KEM) is a cryptographic scheme that produces a symmetric key K and an asymmetric encryption of that key, C. The symmetric key K can then be used in a symmetric encryption scheme, also called a *data encapsulation mechanism* (DEM), to encrypt and transmit long messages. This combination of asymmetric and symmetric encryption is called *hybrid encryption* and is often referred to as the KEM-DEM paradigm, first formalized by Cramer and Shoup [46]. In [130], Shoup showed that by combining a CCA-secure KEM (see Definition 2.33) and a CCA-secure DEM (see [46], Section 7), a hybrid CCA-secure public key encryption scheme (PKE) can be generically obtained. Since CCA-secure DEMs can be efficiently designed [46], when constructing such hybrid encryption schemes focus is placed on the construction of CCA-secure KEMs.

[6] The forward-security property of the BBS generator was first used by Blum and Goldwasser in [27].

Definition 2.32 (KEM)**.** *A* key encapsulation mechanism KEM *consists of three algorithms:*

- $\mathsf{KEM.KG}$*, a probabilistic polynomial-time* key generation algorithm *that on input a security parameter* 1^k*, outputs a pair of public and private keys* (pk, sk)*.*
- $\mathsf{KEM.Enc}$*, a probabilistic polynomial-time* encapsulation algorithm *that takes as input a public key* pk *and outputs a symmetric key* $K \in \mathcal{K}$*, where* $\mathcal{K}$ *is the symmetric key space of the KEM, and an encapsulation (also called ciphertext)* C*.*
- $\mathsf{KEM.Dec}$*, a deterministic polynomial-time* decapsulation algorithm *that takes as input a private key* sk *and an encapsulation* C*, and outputs either a key* $K \in \mathcal{K}$ *or a special symbol* $\perp$*, indicating a failure of decapsulation.*

For correctness we require that for all $k \in \mathbb{N}$ *and all* $(K, C) \leftarrow \mathsf{KEM.Enc}(pk)$*, we have*

$$\Pr\left[\mathsf{KEM.Dec}(sk, C) = K\right] = 1.$$

We see that a KEM allows a sender to generate a symmetric key K which is encapsulated under the receiver's public key pk. The encapsulation, C, is then sent to the receiver, who can recover K by using its private key sk. This can of course be achieved with a public key encryption scheme, but KEMs can be constructed also in different ways.

Chosen-ciphertext security for a KEM is defined in terms of the following IND-CCA experiment (from [88][7]), where an adversary $\mathcal{A}$ is allowed to adaptively query a decapsulation oracle with encapsulations of its choice and obtain the corresponding keys.

Definition 2.33 (IND-CCA security of a KEM)**.** *Let* $\mathsf{KEM} = (\mathsf{KEM.KG}, \mathsf{KEM.Enc}, \mathsf{KEM.Dec})$ *be a key encapsulation mechanism. For any PPT algorithm* $\mathcal{A}$*, we define*

[7] The original definition of IND-CCA security of a KEM is in fact due to [46].

the following experiments:

Experiment $\mathrm{Exp}^{\text{CCA-real}}_{\mathcal{A},\mathsf{KEM}}(1^k)$	Experiment $\mathrm{Exp}^{\text{CCA-rand}}_{\mathcal{A},\mathsf{KEM}}(1^k)$
$(pk, sk) \leftarrow \mathsf{KEM.KG}(1^k)$	$(pk, sk) \leftarrow \mathsf{KEM.KG}(1^k)$
	$K \leftarrow \mathcal{K}$
$(K^*, C^*) \leftarrow \mathsf{KEM.Enc}(pk)$	$(K^*, C^*) \leftarrow \mathsf{KEM.Enc}(pk)$
Return $\mathcal{A}^{\mathsf{KEM.Dec}(sk,\cdot)}(pk, K^*, C^*)$	*Return* $\mathcal{A}^{\mathsf{KEM.Dec}(sk,\cdot)}(pk, K, C^*)$

In the above experiment, the decapsulation oracle $\mathsf{KEM.Dec}(sk, \cdot)$, when queried with an encapsulation $C \neq C^*$, returns either a failure symbol $\perp$ or a key $K \leftarrow \mathsf{KEM.Dec}(sk, C)$; ($\mathsf{KEM.Dec}(sk, \cdot)$ ignores queries $C = C^*$). The advantage of $\mathcal{A}$ in breaking the IND-CCA security of KEM is defined as

$$\mathrm{Adv}^{\mathrm{CCA}}_{\mathcal{A},\mathsf{KEM}}(k) = \left|\Pr[\mathrm{Exp}^{\text{CCA-real}}_{\mathcal{A},\mathsf{KEM}}(1^k) = 1] - \Pr[\mathrm{Exp}^{\text{CCA-rand}}_{\mathcal{A},\mathsf{KEM}}(1^k) = 1]\right|.$$

A KEM scheme is said to be IND-CCA secure if $\mathrm{Adv}^{\mathrm{CCA}}_{\mathcal{A},\mathsf{KEM}}(k)$ is negligible for all polynomial-time adversaries $\mathcal{A}$.

2.3.6 Signature Schemes

Definition 2.34 (Signature schemes)**.** *A signature scheme* SIG *consists of three algorithms:*

- $\mathsf{SIG.KG}$*, a probabilistic polynomial-time* key generation algorithm *that on input* 1^k*, outputs a verification/signing key pair* $(vk, sigk)$*.*
- $\mathsf{SIG.Sign}$*, a probabilistic polynomial-time* signing algorithm *that on input a signing key* $sigk$ *and a message* m*, outputs a signature* σ*.*
- $\mathsf{SIG.Vfy}$*, a deterministic polynomial-time* verification algorithm *that takes as input a verification key* vk*, a message* m *and a signature* σ*, and outputs either* `reject` *or* `accept`*.*

For correctness we require that for all $k \in \mathbb{N}$*, all* $(vk, sigk) \leftarrow \mathsf{SIG.KG}(1^k)$*, all messages* m*, and all* $\sigma \leftarrow \mathsf{SIG.Sign}(sigk, m)$*, we have*

$$\Pr\left[\mathsf{SIG.Vfy}(vk, m, \sigma) = \texttt{accept}\right] = 1.$$

The standard notion of security for a signature scheme is called existential unforgeability under adaptively chosen message attacks (or *unforgeability* for short) [79], which is defined in terms of the following experiment.

Definition 2.35 (Unforgeability)**.** *Let* $\mathsf{SIG} = (\mathsf{SIG.KG}, \mathsf{SIG.Sign}, \mathsf{SIG.Vfy})$ *be a signature scheme. For any PPT algorithm* $\mathcal{A}$*, we define the following experiment:*

$$\begin{array}{l}\text{Experiment } \mathrm{Exp}^{\mathrm{forge}}_{\mathcal{A},\mathsf{SIG}}(1^k)\\ \quad (vk, sigk) \leftarrow \mathsf{SIG.KG}(1^k)\\ \quad (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{SIG.Sign}(sigk,\cdot)}(vk)\\ \quad \textit{If } (\mathsf{SIG.Vfy}(vk, m^*, \sigma^*) = \mathtt{accept} \wedge m^* \neq m_i) \textit{ return } 1 \textit{ else return } 0.\end{array}$$

In the above experiment $\mathcal{A}$ *adaptively issues a polynomial number of queries,* q*, to a signing oracle* $\mathsf{SIG.Sign}(sigk, \cdot)$*. That is, for each* i *(*$1 \leq i \leq q$*),* $\mathcal{A}$ *chooses a message* m_i*, and obtains a signature* σ_i *on that message (i.e.* $\sigma_i = \mathsf{SIG.Sign}(sigk, m_i)$*). A restriction in that experiment is that the forging message* m^* *must be new and has not been signed (i.e.* $m^* \neq m_i$*). The advantage of* $\mathcal{A}$ *in breaking* SIG*'s unforgeability security is defined as*

$$\mathrm{Adv}^{\mathrm{forge}}_{\mathcal{A},\mathsf{SIG}}(k) = \Pr\left[\mathrm{Exp}^{\mathrm{forge}}_{\mathcal{A},\mathsf{SIG}}(1^k) = 1\right].$$

A signature scheme is said to be unforgeable *if* $\mathrm{Adv}^{\mathrm{forge}}_{\mathcal{A},\mathsf{SIG}}(k)$ *is negligible for all polynomial-time adversaries* $\mathcal{A}$*.*

A stronger definition of unforgeability can be obtained if we relax Definition 2.35 by allowing the adversary $\mathcal{A}$, in the security experiment, to output a new valid signature on a message that could have been signed before. That is, $\mathcal{A}$ wins the security experiment if it outputs a pair (m^*, σ^*) such that $\mathsf{SIG.Vfy}(vk, m^*, \sigma^*) = \mathtt{accept} \wedge (m^*, \sigma^*) \neq (m_i, \sigma_i)$). This notion is called *strong existential unforgeability under adaptively chosen message attacks* (or *strong unforgeability* for short).

In this thesis, one of the cryptographic primitives we use is a *strong one-time signature* (strong-OTS) scheme [109]. Informally, a strong-OTS scheme is a signature scheme that requires each signing key to be used only once for each message to be signed. The security experiment is just like the one for strong unforgeability, except that for strong-OTS, given a verification key, the adversary is only allowed to issue at most one query to the signing oracle before producing a forgery on a message that could have been signed by the signing oracle. More formally, consider the following definition.

Definition 2.36 (Strong-OTS)**.** *Let* $\mathsf{OTS} = (\mathsf{OTS.KG}, \mathsf{OTS.Sign}, \mathsf{OTS.Vfy})$ *be a signature scheme. For any PPT algorithm* $\mathcal{A}$*, we define the following experiment:*

Experiment $\mathrm{Exp}^{\text{S-OTS}}_{\mathcal{A},\mathsf{OTS}}(1^k)$
$(vk, sigk) \leftarrow \mathsf{OTS.KG}(1^k)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{OTS.Sign}(sigk,\cdot)}(vk)$
If $(\mathsf{OTS.Vfy}(vk, m^*, \sigma^*) = \mathtt{accept} \wedge (m^*, \sigma^*) \neq (m, \sigma))$ *return* 1 *else return* 0.

In the above experiment $\mathcal{A}$ *is allowed to issue* at most one *query to a signing oracle* $\mathsf{SIG.Sign}(sigk, \cdot)$*. That is,* $\mathcal{A}$ *can choose a message* m*, and obtain a signature* σ *on that message. Note that* $\mathcal{A}$ *is successful in the above experiment even if the forging message* m^* *that it outputs is the same as the message* m *that it may have queried to the signing oracle* $\mathsf{OTS.Sign}(sigk, \cdot)$*, as long as* $\sigma^* \neq \sigma$*. The advantage of* $\mathcal{A}$ *in breaking* OTS*'s strong one-time security is defined as*

$$\mathrm{Adv}^{\text{S-OTS}}_{\mathcal{A},\mathsf{OTS}}(k) = \Pr\left[\mathrm{Exp}^{\text{S-OTS}}_{\mathcal{A},\mathsf{OTS}}(1^k) = 1\right].$$

A signature scheme is said to be strongly one-time secure if $\mathrm{Adv}^{\text{S-OTS}}_{\mathcal{A},\mathsf{OTS}}(k)$ *is negligible for all polynomial-time adversaries* $\mathcal{A}$*.*

Similarly, a regular one-time signature (rather than strong) scheme can be defined by restricting the adversary $\mathcal{A}$ in the above definition to output a signature forgery on a message different from the one (possibly) queried to the signing oracle.

Part I

Non-Interactive Key Exchange

Chapter 3

Non-Interactive Key Exchange (NIKE)

Contents

In this chapter we systematically study non-interactive key exchange schemes in the public key setting (NIKE). We provide different security models for this cryptographic primitive and explore the relationships between them. We then give constructions for secure NIKE in the standard model and in the random oracle model. Furthermore, we also study the relationship between NIKE and public key encryption (PKE), showing that a secure NIKE scheme can be generically converted into an IND-CCA secure PKE scheme. Most of the content of this chapter appears in [63], which is joint work with Dennis Hofheinz, Eike Kiltz and Kenneth G. Paterson.

3.1 Introduction

Non-interactive key exchange (NIKE) is a cryptographic primitive which enables two users, who know each others' public keys, to agree on a symmetric shared key without requiring any interaction. The canonical example of a NIKE scheme can be found in the seminal paper by Diffie and Hellman [53]: let $\mathbb{G}$ be a group of prime order p with generator g, and assume Alice has public key $g^x \in \mathbb{G}$ and private key $x \in \mathbb{Z}_p$, while Bob has public key $g^y \in \mathbb{G}$ and private key $y \in \mathbb{Z}_p$. Then Alice and Bob can both compute the value $g^{xy} \in \mathbb{G}$ without exchanging any messages. More properly, Alice and Bob should hash this key together with their identities in order to derive a symmetric key $\mathsf{H}(\texttt{Alice}, \texttt{Bob}, g^{xy})$.

This example encapsulates in a nutshell all the basic features required of a NIKE scheme: users should agree on some common parameters (p, $\mathbb{G}$ and g here), then create their key pairs. Once these are computed and the public keys distributed, any pair of users can set up a shared key without further exchange of messages. The security properties desired of NIKE are, informally at least, clear: compromise of one user's private key should not affect the security of shared keys between pairs of uncorrupted users; compromise of a shared key between a pair of users should not undermine the security of shared keys between other pairs of users. In practice, the public keys will be certified, and consideration needs to be given to modelling the key registration process.

NIKE has real-world applications. In wireless and sensor networks, conserving battery power is a prime concern, and so the energy cost of communication must be minimised. Thus using key establishment methods that minimise the number of bits that need to be transmitted is of fundamental importance. In particular, when faced with a jamming adversary, reducing the total number of rounds of interaction needed to establish a key is particularly helpful. NIKE is an excellent option in solving this problem, since a key can be established with minimal communication and interaction: assuming the public keys are pre-distributed, all that is needed is an exchange of identifiers for those keys, and often this exchange must take place anyway, in order to establish communications. A recent paper [42] gives a detailed evaluation of the energy costs of interactive and non-interactive key exchange protocols in the ID-based and public-key settings for wireless communications with a

jamming adversary, demonstrating that significant energy savings can be made by adopting a non-interactive approach to key establishment. Its non-interactive nature makes NIKE an abstract building block that is *qualitatively* different from interactive key exchange: e.g. to achieve deniable authentication, [56] explicitly requires a *non-interactive* key exchange. But NIKE can also be used as a basis for *interactive* key exhange [13, 34, 117]: for example, in [34] the authors use the shared key in a MAC to authenticate an exchange of ephemeral Diffie-Hellman values. The TLS Handshake Protocol [52], which is the *de facto* protocol of choice for secure communication on the Internet, allows as an option a Diffie-Hellman key derived from static, long-term public keys to be used as its pre-master-secret, from which all other keys in the protocol are derived.[1] Finally, NIKE can be used to build very simple non-interactive designated verifier signature schemes [91], again using the shared key in a MAC to authenticate messages. Thus NIKE appears in various guises throughout the literature.

Despite its appearing in the very first paper on public key cryptography, the NIKE primitive has so far received scant attention as a cryptographic primitive in its own right. Bernstein [22] proposed an efficient NIKE scheme in the elliptic-curve setting and sketched a security model for NIKE. Cash, Kiltz and Shoup (CKS) [40] provided a formal security model for NIKE and analysed the Diffie-Hellman-based scheme above, as well as a twinned variant of it, in the Random Oracle Model (ROM). In the ID-based setting, Dupont and Enge [57] introduced a security model for NIKE and analysed the Sakai-Ohgishi-Kasahara (SOK) scheme [122] in this model. In a follow-up work, Paterson and Srinivasan [112] provided a more refined security model and explored the connections between ID-based NIKE and Identity-Based Encryption (IBE). Gennaro *et al.* [73] developed NIKE protocols for the hierarchical ID-based setting. All of these papers use the ROM.

3.1.1 Our Contributions

This chapter is dedicated to a systematic study of NIKE in the public key setting. We provide: security models for NIKE and their relationships; constructions for secure

[1] However, the long-term public keys are actually exchanged during the protocol, so strictly speaking the protocol is interactive even though the key can be seen as arising from a non-interactive exchange.

NIKE in the random oracle model and in the standard model in the challenging setting where the adversary can introduce arbitrary public keys into the system; and a construction for IND-CCA secure public key encryption (PKE) from any secure NIKE. Let us expand on each of these contributions in turn.

Models: It would seem that definitions and security models for interactive key exchange (e.g. [17, 21, 37, 129]) could provide a natural starting point for formalising NIKE. However, here we take the CKS definition [40] for NIKE as our starting point. One reason for using a case-tailored NIKE definition is simplicity: existing security models for interactive key exchange give considerable attention to properties which are irrelevant in the NIKE setting. (For instance, forward security, multiple sessions, and in particular the pairing of sessions play no role in a non-interactive setting.) Another reason for a case-tailored NIKE definition is that we can focus on adversarial key registration queries; these are usually only implicitly [37] (or not at all [17, 21]) considered in the standard models for interactive key exchange.[2] However, in our setting, adversarial key registration poses the main technical obstacle to achieve NIKE security, as we will explain below.

The CKS security model for NIKE uses an indistinguishability and game-based approach to define security, with the adversary being required to distinguish real from random keys in responses to its test queries. The model does allow the adversary to register public keys of its choice in the system and then to make queries for the shared keys between these "corrupted" users and honest (non-adversarially controlled) users, so-called `corrupt reveal` queries. This translates in the real world to minimising the assumptions made about certification procedures followed by the Certification Authority (CA) in the PKI supporting the NIKE: it means that the CA is not assumed to check that a public key submitted for certification has not been submitted before, and does not check that the user submitting the public key knows the corresponding private key. The model for NIKE in [40] is similar to, and presumably inspired by, the early work of Shoup [129] on interactive key exchange, where capturing so-called PKI attacks, also known as rogue-key attacks, was intrinsic to the security modelling. This modelling approach is referred to elsewhere in the literature as the *plain* setting (see [16, 118] and the references therein) or the

[2] We mention that some security analyses (e.g. [101]) and Shoup's security model [129] do explicitly consider adversarial key registration queries.

bare PKI setting [56]. The CKS model is certainly more challenging than settings where proofs of knowledge or proofs of possession of private keys are assumed to be given during registration, or where the adversary must reveal its private key directly (as with the knowledge of private key assumption used in [30, 103]). However, the CKS model has some shortcomings: the adversary is not allowed to directly query for the shared keys held between pairs of honest users, but instead only gets to see real or random values for these via test queries. Moreover the model does not allow an adversary to query for the private keys of honestly registered users.

Therefore, as a necessary precursor to the further development of NIKE, we start by exploring different models for NIKE and their relationships (Section 3.2). In summary, we introduce three new security models for NIKE and show that they are all polynomially equivalent to one another and to the original CKS model from [40]. One of our models, the m-CKS-heavy model, augments the CKS model and effectively allows all conceivable queries, without allowing the adversary to win trivially. Another of our models, CKS-light, allows only two honest users, no corruption of honest users, and a single test query. Thus it is particularly simple and easy to use when analysing specific NIKE schemes; moreover our results showing equivalence between the models ensure that security in this model implies security in the stronger m-CKS-heavy model.

We stress that all these models allow the adversary to register public keys of its choice in the system, so are in the plain setting, or *dishonest key registration* (DKR) setting. However, for completeness, we also briefly consider the *honest key registration* (HKR) setting in which the adversary cannot register keys on its own. It is easy to see that the HKR setting provides strictly weaker security guarantees than our default security setting with dishonest key registration. For instance, the already mentioned Diffie-Hellman NIKE scheme *without hashing* (such that shared keys are of the form g^{xy}) can be shown secure in the HKR setting under the Decisional Diffie-Hellman assumption, but is easily seen to be completely insecure in our default setting.[3]

[3]Concretely, since shared keys do not depend on user identities in the unhashed DH-NIKE, an adversary $\mathcal{A}$ can (a) register the key g^x of an honest user Alice as its own key, and (b) ask for the shared key between $\mathcal{A}$ and another honest user Bob with key g^y. This immediately yields the shared key g^{xy} between Alice and Bob. Because of the homomorphic properties of the DH-NIKE, a simple modification of this attack also works if $\mathcal{A}$ is not allowed to register keys of existing users. A similar attack applies to the scheme in [22].

Constructions for NIKE: In Section 3.3, we give two concrete constructions for NIKE schemes meeting our CKS-light security definition, and hence secure in our stronger m-CKS-heavy model (*with* dishonest key registration). Our first NIKE scheme is provably secure in the standard model and combines a specific weak Programmable Hash Function [89] (see also Section 2.3.2), whose output lies in a pairing group, and a Chameleon hash function. This enables the simulation in our security proof for the scheme to handle the tricky queries for shared keys involving an honestly generated public key and an adversarially chosen public key. We also make use of the pairing to provide a means of checking that public keys coming from the adversary are in some sense well-formed. We work with asymmetric pairings for efficiency at high security levels (and because it does not add any real complexity to the description of our scheme). The scheme's security relies on a natural variant of the Decisional Bilinear Diffie-Hellman (DBDH) assumption for the asymmetric setting.

Our second scheme is provably secure under the factoring assumption in the Random Oracle Model (ROM) and uses ideas from [87] to analyse the hashed Diffie-Hellman scheme, where keys are of the form $\mathsf{H}(\mathtt{Alice}, \mathtt{Bob}, g^{xy})$, in the group of signed quadratic residues (see Definition 2.14). We note that closely related schemes were analysed in [40], but in different groups and under different assumptions. Specifically, a twinned version of the scheme was proved secure under the CDH assumption, while it is stated that the hashed Diffie-Hellman scheme is secure under the SDH assumption. We remark that the latter claim of [40] is problematic. Concretely, the SDH assumption is not (directly) sufficient to show that the basic Diffie-Hellman scheme is secure. Namely, the corresponding security reduction requires *two* DDH oracles – one for each of the two users sharing the key on which the adversary wants to be challenged – while the SDH assumption supplies only one. Certainly this problem could be solved instead by appealing to a suitable gap-DH assumption. We show how to overcome this problem in the group of signed quadratic residues without the need to rely on a gap assumption.

From NIKE to PKE: In Section 3.4, we explore the connections between NIKE and public key encryption (PKE). That such connections exist should not be too much of a surprise: it is folklore that the ElGamal encryption scheme [71] can be seen as arising from the Diffie-Hellman NIKE scheme by making the sender's key

pair (g^x, x) ephemeral and using the receiver's public key g^y to create the basis for a shared key g^{xy}. In fact, a simple transformation shows that every NIKE that is secure in the simpler HKR setting can be turned into a public key encryption scheme that is secure against chosen-plaintext attacks (IND-CPA secure). Similar connections were explored in the ID-based setting in [112].

In our default setting with dishonest key registration (the DKR setting), we provide a simple, generic construction for PKE from NIKE that is also in the spirit of the original Diffie-Hellman–to–ElGamal conversion. The construction takes a NIKE scheme that is secure in our CKS-light model (with dishonest key registration) and a strongly one-time secure signature scheme as inputs, and produces from these components a Key Encapsulation Mechanism (KEM) that we prove to be IND-CCA secure. A secure PKE from such a KEM can be obtained using standard results. At a high level, the key pair for the KEM is a randomly generated key pair (pk, sk) from the NIKE scheme, ciphertexts are also randomly generated public keys pk' from the NIKE scheme (together with a one-time signature that binds the public key to an identity), while the encapsulated key is the shared key computed from sk' and pk; the receiver computes the same key from sk and pk', assuming the one-time signature verifies. In order to prove the KEM to be IND-CCA secure, we exploit the presence of `corrupt reveal` queries in the NIKE security model in an essential way to handle certain decapsulation queries. The resulting KEM is almost as efficient as the underlying NIKE scheme. In the HKR setting, the same transformation (only without one-time signatures) shows that CKS-light security of the NIKE scheme implies IND-CPA security of the resulting PKE scheme.

In Section 3.5, we provide a closely-related conversion that starts with a secure NIKE scheme satisfying a simplified definition (basically, it omits all consideration of identities) and produces an IND-CCA secure KEM without using one-time signatures. This results in more efficient KEMs. We can apply the conversion with a simplified version of the concrete NIKE scheme from Section 3.3.1 to obtain an attractive KEM that is IND-CCA secure in the standard model under our asymmetric variant of the DBDH assumption. This KEM is comparable in performance to the scheme proposed in [35] and neatly illustrates the utility of NIKE as a primitive, as well as its connections with other classical public key primitives.

The fact that secure NIKE implies IND-CCA-secure PKE, one of the most important primitives in cryptography, illustrates the fundamental role and utility of NIKE. We believe that this connection should encourage further research on the topic.

3.2 Non-Interactive Key Exchange

Following [40], we formally define a Non-Interactive Key Exchange (NIKE) scheme in the public key setting, NIKE, as a collection of three algorithms: NIKE.Setup, NIKE.KeyGen and NIKE.SharedKey, together with an identity space $\mathcal{ID}$ and a shared key space $\mathcal{SHK}$. Note that identities in the scheme and security model are merely used to track which public keys are associated with which users.

- NIKE.Setup: On input 1^k, outputs *params*, a set of system parameters.
- NIKE.KeyGen: On input *params* and an identity $id \in \mathcal{ID}$, outputs a public key/private key pair (pk, sk). This algorithm is probabilistic and can be executed by any user.
- NIKE.SharedKey: On input an identity $id_1 \in \mathcal{ID}$ and a public key pk_1 along with another identity $id_2 \in \mathcal{ID}$ and a private key sk_2, outputs either a shared key in $\mathcal{SHK}$ for the two identities, or a failure symbol $\perp$. This algorithm is assumed to always output $\perp$ if $id_1 = id_2$.

For correctness, we require that, for any pair of identities id_1, id_2, and corresponding key pairs (pk_1, sk_1) and (pk_2, sk_2), algorithm NIKE.SharedKey satisfies the constraint:

$$\mathsf{NIKE.SharedKey}(id_1, pk_1, id_2, sk_2) = \mathsf{NIKE.SharedKey}(id_2, pk_2, id_1, sk_1).$$

3.2.1 Definitions of Security for NIKE

Cash, Kiltz and Shoup [40] proposed a security model for NIKE schemes in the public key setting, denoted here by the CKS model. This model abstracts away all considerations concerning certification and PKI in a particularly nice way. It allows

an adversary to obtain honestly generated public keys, but also to then associate such public keys with other identities, and to register dishonestly generated public keys (for which the adversary need not know the corresponding private keys). This dishonest key registration (DKR) setting models a PKI where minimal assumptions are made about the actions of the Certification Authority (CA): the CA is not assumed to check that a public key has not been previously registered to another user, and does not demand a proof of knowledge or possession of the private key when issuing a certificate on a public key. This conservative approach to modelling is fully appropriate given the great diversity of how CAs operate in the real world. The model can be seen as a natural adaptation of the approach of Shoup [129] for modelling interactive key exchange to the NIKE setting and is analogous to the plain setting studied in [16, 118].

However, there are some obvious omissions from the model in [40], including the ability of an adversary to learn private keys associated with honestly generated public keys and the ability of a user to directly learn the key shared between two honest users in the system (which could be possible, for example, because of cryptanalysis of a scheme making use of the shared key). Equivalent queries in the ID-based setting were permitted in the model introduced in [112].

For this reason, we augment the original CKS model with the "missing" queries, introducing the m-CKS-heavy model. We also introduce two further models, the CKS-heavy and CKS-light models. These differ from m-CKS-heavy and the original CKS model only in the numbers and types of query that the adversary is allowed to make. Next we present in detail the m-CKS-heavy model. Then in Table 3.1 we summarize the differences between these security models in the DKR setting.

The m-CKS-heavy **model:** Our model is stated in terms of a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. In this game, $\mathcal{C}$ takes as input the security parameter 1^k, runs algorithm $\mathsf{NIKE.Setup}$ of the NIKE scheme and gives $\mathcal{A}$ *params*. The challenger takes a random bit b and answers oracle queries for $\mathcal{A}$ until $\mathcal{A}$ outputs a bit $\hat{b}$. The challenger answers the following types of queries for $\mathcal{A}$:

- `register honest user`: $\mathcal{A}$ supplies an identity $id \in \mathcal{ID}$. The challenger $\mathcal{C}$ first obtains a pair of public key and private key by running $(pk, sk) \leftarrow$

$\mathsf{NIKE.KeyGen}(params, id)$. It then records the tuple $(honest, id, pk, sk)$ and returns pk to $\mathcal{A}$.

- `register corrupt user`: In this type of query, $\mathcal{A}$ supplies both an identity $id \in \mathcal{ID}$ and a public key pk. The challenger $\mathcal{C}$ records the tuple $(corrupt, id, pk, \perp)$. We stress that $\mathcal{A}$ may make multiple "`register corrupt user`" queries for the same id during the experiment. In that case, only the most recent $(corrupt, id, pk, \perp)$ entry is kept.

- `extract`: Here $\mathcal{A}$ supplies an identity id that was registered as an honest user. The challenger looks for a tuple $(honest, id, pk, sk)$ containing id and returns sk to $\mathcal{A}$.

- `reveal`: Here $\mathcal{A}$ supplies a pair of registered identities id_1, id_2, subject only to the restriction that at least one of the two identities was registered as *honest*. Without loss of generality, assume id_1 was registered as *honest*. The challenger $\mathcal{C}$ runs $\mathsf{NIKE.SharedKey}(id_2, pk_2, id_1, sk_1)$ and returns the result to $\mathcal{A}$. Note that here the adversary is allowed to make `reveal` queries between two users that were originally registered as honest users. We denote by `honest reveal` the queries involving two honest users and by `corrupt reveal` the queries involving an honest user and a corrupt user.

- `test`: $\mathcal{A}$ supplies two distinct identities id_1, id_2 that were both registered as honest. The challenger returns $\perp$ if $id_1 = id_2$. Otherwise, it uses the bit b to answer the queries. If $b = 1$, the challenger runs $\mathsf{NIKE.SharedKey}(id_1, pk_1, id_2, sk_2)$ and returns the result to $\mathcal{A}$. If $b = 0$, the challenger generates a random key from $\mathcal{SHK}$, records it for later, and returns that key to the adversary. In this case, to keep things consistent, the challenger returns the same random key for the pair id_1, id_2 every time $\mathcal{A}$ queries for their shared key.

$\mathcal{A}$'s queries may be made adaptively and are arbitrary in number. To prevent trivial wins for the adversary, no query to the `reveal` oracle is allowed on any pair of identities selected for `test` queries (in either order), and no `extract` query is allowed on any of the identities involved in `test` queries. Also, we demand that no identity registered as corrupt can later be the subject of a `register honest user` query, and vice versa.

Model	`register honest`	`register corrupt`	`extract`	`honest reveal`	`corrupt reveal`	`test`
CKS-light	2	✓	✗	✗	✓	1
CKS	✓	✓	✗	✗	✓	✓
CKS-heavy	✓	✓	✓	✓	✓	1
m-CKS-heavy	✓	✓	✓	✓	✓	✓

Table 3.1: Types of queries for different security models in the dishonest key registration (DKR) PKI model (aka plain/bare model). Notation: ✓means that an adversary is allowed to make an arbitrary number of queries; ✗ means that no queries can be made; integers represent the number of queries allowed to an adversary.

When the adversary finally outputs $\hat{b}$, it wins the game if $\hat{b} = b$. For an adversary $\mathcal{A}$, we define its advantage in this security game as:

$$\mathrm{Adv}^{\text{m-CKS-heavy}}_{\mathcal{A},\mathsf{NIKE}}(k, q_H, q_C, q_E, q_{HR}, q_{CR}, q_T) = 2\,|\Pr[\hat{b} = b] - 1/2|,$$

where q_H, q_C, q_E, q_{HR}, q_{CR} and q_T are, respectively, the numbers of `register honest user`, `register corrupt user`, `extract`, `honest reveal`, `corrupt reveal` and `test` queries made by $\mathcal{A}$. We say that a NIKE scheme is $(t, \epsilon, q_H, q_C, q_E, q_{HR}, q_{CR}, q_T)$-secure in the m-CKS-heavy model if there is no adversary with advantage at least ϵ that runs in time t and makes at most q_H `register honest user` queries, etc. Informally, we say that a NIKE scheme is m-CKS-heavy secure if there is no efficient adversary having non-negligible advantage in k, where efficient means that the running time and numbers of queries made by the adversary are bounded by polynomials in k. For simplicity we may also use $\mathrm{Adv}^{\text{m-CKS-heavy}}_{\mathcal{A},\mathsf{NIKE}}(k)$ to denote the advantage of $\mathcal{A}$, in the m-CKS-heavy model, against a NIKE scheme NIKE.

Comparing the models: Table 3.1 outlines the properties of all our security models in the DKR setting, and the CKS model, in terms of restrictions on the queries that can be made by the adversary. It is apparent that the m-CKS-heavy model is the strongest model in that table. It differs from the CKS-heavy model only in allowing multiple `test` queries. The m-CKS-heavy model represents a strengthening of the original CKS model by allowing `extract` and `honest reveal` queries, whereas the CKS model only allows the adversary to gain information about honestly generated shared keys via `test` queries. The CKS-light model is the simplest of all, involving only two honestly registered identities, removing the `extract` and `honest reveal` queries, and allowing only a single `test` query.

In Theorem 3.1 we prove our strongest model, the m-CKS-heavy model, to be polynomially equivalent to our weakest model, the CKS-light model. Then, it follows that all security models are also polynomially equivalent. Thus, while the m-CKS-heavy is our preferred model, it suffices to analyse schemes in the CKS-light model if one is not overly concerned about concrete security. (Note that the security reductions are not tight. It is an interesting open problem to either prove tighter relations between the models, or to prove that such results are not possible.)

Theorem 3.1 (m-CKS-heavy $\Leftrightarrow$ CKS-light)**.** *The* m-CKS-heavy *and* CKS-light *security models are polynomially equivalent. More specifically, for any NIKE scheme* NIKE*, we have that:*

- *for any adversary* $\mathcal{B}$ *against* NIKE *in the* CKS-light *game, there is an adversary* $\mathcal{A}$ *that breaks* NIKE *in the* m-CKS-heavy *game with*

$$\mathrm{Adv}^{\text{m-CKS-heavy}}_{\mathcal{A},\mathsf{NIKE}}(k, 2, q_C, 0, 0, q_{CR}, 1) = \mathrm{Adv}^{\text{CKS-light}}_{\mathcal{B},\mathsf{NIKE}}(k, q_C, q_{CR}),$$

- *for any adversary* $\mathcal{A}$ *against* NIKE *in the* m-CKS-heavy *game, there is an adversary* $\mathcal{B}$ *that breaks* NIKE *in the* CKS-light *game with*

  $\mathrm{Adv}^{\text{CKS-light}}_{\mathcal{B},\mathsf{NIKE}}(k, q_C, q'_{CR}) \geq (2/q_T q_H^2) \cdot \mathrm{Adv}^{\text{m-CKS-heavy}}_{\mathcal{A},\mathsf{NIKE}}(k, q_H, q_C, q_E, q_{HR}, q_{CR}, q_T)$,

  where $q'_{CR} \leq q_{CR}$*.*

Proof. Clearly, security in the sense of the m-CKS-heavy model implies security in the sense of the CKS-light model, since the latter model is a limited case of the former.

Here we prove the second reduction, namely that if a NIKE scheme NIKE is secure in the CKS-light model, then it is also secure in the m-CKS-heavy model. We assume that there is an adversary $\mathcal{A}$ that breaks a NIKE scheme in the m-CKS-heavy model with advantage $\mathrm{Adv}^{\text{m-CKS-heavy}}_{\mathcal{A},\mathsf{NIKE}}(k, q_H, q_C, q_E, q_{HR}, q_{CR}, q_T)$.

The hybrid argument technique. We consider a sequence of games $G_0, G_1, \ldots, G_{q_T}$, all defined over the same probability space. Starting with the actual adversarial game G_0, with respect to an adversary $\mathcal{A}$ in the m-CKS-heavy model when $b = 0$, we make slight modifications between successive games, in such a way that $\mathcal{A}$'s view is indistinguishable among the games.

Game G_0: actual adversarial game when $b = 0$. This is the original game as defined in the m-CKS-heavy model when $b = 0$. All `test` queries will be answered with randomly chosen values from the shared key space $\mathcal{SHK}$.

Game G_ι ($1 \leq \iota \leq q_T - 1$): hybrid games. This game is identical to game $G_{\iota-1}$, except that the ι-th `test` query, say on a pair of honest users id_i, id_j, is answered with the actual real shared key, K_{id_i,id_j}, between those users.

Game G_{q_T}: actual adversarial game when $b = 1$. This is the original game as defined in the m-CKS-heavy model when $b = 1$. All `test` queries will be answered with the real shared keys (computed via the NIKE.SharedKey algorithm) associated with the two users involved in each query.

Let $\mathcal{A}(G_\iota)$ denote adversary $\mathcal{A}$ playing game G_ι. We see that $\mathcal{A}$ can distinguish games G_0 and G_{q_T} with advantage

$$\begin{aligned}&\mathrm{Adv}^{\text{m-CKS-heavy}}_{\mathcal{A},\mathsf{NIKE}}(k, q_H, q_C, q_E, q_{HR}, q_{CR}, q_T)\\ &\quad = \left|\Pr[\mathcal{A}(G_{q_T}) = 1] - \Pr[\mathcal{A}(G_0) = 1]\right|.\end{aligned}$$

Note that G_ι and $G_{\iota+1}$ ($0 \leq \iota < q_T$) differ in only one single `test` query. According to the *hybrid argument*, if $\mathcal{A}$ can distinguish between G_0 and G_{q_T} with non-negligible probability, then for some $0 \leq \iota < q_T$ it can also distinguish G_ι and $G_{\iota+1}$ with non-negligible probability. We show that if this is the case, then we can construct an adversary $\mathcal{B}$ in the CKS-light model with advantage related to $\mathcal{A}$'s advantage by a polynomial factor.

The idea. In order to simulate the m-CKS-heavy game for $\mathcal{A}$, $\mathcal{B}$ picks $\iota \leftarrow \{0, \ldots, q_T - 1\}$, guessing that $\mathcal{A}$ can distinguish games G_ι and $G_{\iota+1}$. Then it guesses which users will be involved in the $(\iota + 1)$-st `test` query made by $\mathcal{A}$ (more precisely, $\mathcal{B}$ guesses that these users will be the I-th and J-th users registered as *honest*).

Adversary $\mathcal{B}$ plays the CKS-light security game with challenger $\mathcal{C}$ and acts as a challenger for $\mathcal{A}$. On input the security parameter 1^k, $\mathcal{C}$ computes $params \leftarrow \mathsf{NIKE.Setup}(1^k)$ and gives $params$ to $\mathcal{B}$. The challenger $\mathcal{C}$ then takes a bit b and answers oracle queries for $\mathcal{B}$ until $\mathcal{B}$ outputs a bit $\hat{b}$.

Let q_H be a bound on the number of honest users registered by $\mathcal{A}$ in the course of its

attack and let q_T be a bound on the number of `test` queries made by $\mathcal{A}$. Without loss of generality, we assume that $\mathcal{A}$ does not make the same `test` query more than once. The CKS-light adversary $\mathcal{B}$ chooses an index $\iota \leftarrow \{0, \ldots, q_T - 1\}$ and two distinct indices $I, J \leftarrow \{1, \ldots, q_H\}$. It then gives *params* to $\mathcal{A}$ and answers queries to $\mathcal{A}$ in the following manner:

- `register honest user` (id):

  If id is the I-th or J-th distinct user involved in such a query, $\mathcal{B}$ sets $id_I = id$ or $id_J = id$ as appropriate. It adds id_I (resp. id_J) to a list Λ_T. Adversary $\mathcal{B}$ then makes the same query to $\mathcal{C}$, which obtains $(pk, sk) \leftarrow \mathsf{NIKE.KeyGen}(params, id)$, records $(honest, id, pk, sk)$, and returns pk to $\mathcal{B}$. $\mathcal{B}$ gives pk to $\mathcal{A}$. Otherwise, if id is not the I-th or J-th distinct user involved in a `register honest user` query made by $\mathcal{A}$, then $\mathcal{B}$ computes $(pk, sk) \leftarrow \mathsf{NIKE.KeyGen}(params, id)$ and sends pk to $\mathcal{A}$. $\mathcal{B}$ stores $(honest, id, pk, sk)$ in a list Λ_{hon}.

- `register corrupt user` (id, pk):

  Here $\mathcal{A}$ supplies a public key pk along with an identity id that has not been subject to a `register honest user` query. $\mathcal{B}$ forwards the `register corrupt user` query to $\mathcal{C}$, which will record $(corrupt, id, pk, \perp)$. $\mathcal{B}$ stores $(corrupt, id, pk, \perp)$ in a list Λ_{cor}.

- `extract` (id):

  If $id \notin \Lambda_T$, $\mathcal{B}$ finds $(honest, id, pk, sk)$ in Λ_{hon} and gives sk to $\mathcal{A}$. Otherwise, if $id \in \Lambda_T$, $\mathcal{B}$ aborts the simulation.

- `corrupt reveal` (id_i, id_j):

  Here $\mathcal{A}$ supplies two identities id_i and id_j, where either id_i or id_j is an honest user. Without loss of generality, let us assume that id_j is the honest user. $\mathcal{B}$ checks if $id_j \in \Lambda_{\text{hon}}$. If $id_j \notin \Lambda_{\text{hon}}$ (this means that $id_j \in \Lambda_T$), $\mathcal{B}$ makes the same query to $\mathcal{C}$, obtaining K_{id_i, id_j}, the shared key between id_i and id_j. $\mathcal{B}$ returns this value to $\mathcal{A}$. Now, if $id_j \in \Lambda_{\text{hon}}$, $\mathcal{B}$ finds sk_j, then it finds pk_i in Λ_{cor} and computes $K_{id_i, id_j} \leftarrow \mathsf{NIKE.SharedKey}(id_i, pk_i, id_j, sk_j)$. $\mathcal{B}$ then returns K_{id_i, id_j} to $\mathcal{A}$.

- `honest reveal` (id_i, id_j):

Here $\mathcal{A}$ supplies two identities id_i and id_j, both registered as honest users. If $\{id_i, id_j\} \subseteq \Lambda_T$, $\mathcal{B}$ aborts. Otherwise, at least one of the identities must be in Λ_{hon}. Without loss of generality, assume that $id_i \in \Lambda_{\text{hon}}$. $\mathcal{B}$ finds the private key sk_i in Λ_{hon}, then computes $K_{id_i,id_j} \leftarrow \mathsf{NIKE.SharedKey}(id_j, pk_j, id_i, sk_i)$ and returns K_{id_i,id_j} to $\mathcal{A}$.

- `test` (id_i, id_j):

  The way $\mathcal{B}$ answers `test` queries depends on the number of such queries issued by $\mathcal{A}$.

  - the first ι `test` queries, $\mathcal{B}$ answers with the actual shared keys associated with the corresponding users involved in those queries. In order to do this, $\mathcal{B}$ checks if $\{id_i, id_j\} \subseteq \Lambda_T$. If so, it aborts. Otherwise, at least one of the identities must be in Λ_{hon}. Without loss of generality, assume that $id_i \in \Lambda_{\text{hon}}$. Now $\mathcal{B}$ retrieves the private key sk_i from Λ_{hon}, then computes $K_{id_i,id_j} \leftarrow \mathsf{NIKE.SharedKey}(id_j, pk_j, id_i, sk_i)$ and returns K_{id_i,id_j} to $\mathcal{A}$.
  - if this is the $(\iota+1)$-st `test` query, $\mathcal{B}$ checks if $\{id_i, id_j\} \subseteq \Lambda_T$. If not, it aborts. If $\{id_i, id_j\} \subseteq \Lambda_T$, $\mathcal{B}$ makes the same `test` query to $\mathcal{C}$, receiving a value α. $\mathcal{B}$ gives α to $\mathcal{A}$.
  - all other `test` queries $\mathcal{B}$ answers to $\mathcal{A}$ with random values from $\mathcal{SHK}$.

This completes the description of $\mathcal{B}$'s simulation. Whenever $\mathcal{A}$ outputs a bit $\hat{b}$, $\mathcal{B}$ outputs the same bit. Now, if α is the actual shared key computed via the $\mathsf{NIKE.SharedKey}$ algorithm and using as inputs the private key of one the users involved in the $(\iota+1)$-st `test` query and the public key of the other user, then $\mathcal{A}$ was playing game $G_{\iota+1}$. Otherwise, if α is a random value, $\mathcal{A}$ was playing game G_ι.

Let G'_0 and G'_1 be the CKS-light games played by $\mathcal{B}$ when $b = 0$ and $b = 1$, respectively. Let `F` denote the event that $\mathcal{B}$ correctly guessed the users involved in the $(\iota+1)$-st `test` query (this would mean that the users are id_I and id_J). If `F` occurs then $\mathcal{B}$ simulates the m-CKS-heavy game correctly. Now we assess $\mathcal{B}$'s success probability. It is easy to see that $\Pr[\mathtt{F}] = 1/\binom{q_H}{2} \geq 2/q_H^2$.

Averaging the probability that $\mathcal{B}$ outputs 1 over the random choice of ι, we have:

$$\Pr[\mathcal{B}(G'_1) = 1] = \frac{1}{q_T} \Pr[\mathtt{F}] \sum_{\iota=0}^{q_T-1} \Pr[\mathcal{A}(G_{\iota+1}) = 1]$$

Model	register honest	register corrupt	extract	honest reveal	corrupt reveal	test
HKR CKS-light	2	✗	✗	✗	✗	1
HKR CKS	✓	✗	✗	✗	✗	✓
HKR CKS-heavy	✓	✗	✓	✓	✗	1
HKR m-CKS-heavy	✓	✗	✓	✓	✗	✓

Table 3.2: Types of queries for different security models in the honest key registration (HKR) PKI model.

and

$$\Pr[\mathcal{B}(G'_0) = 1] = \frac{1}{q_T} \Pr[\mathtt{F}] \sum_{\iota=0}^{q_T-1} \Pr[\mathcal{A}(G_\iota) = 1].$$

Therefore it follows that

$$\begin{aligned}
\mathrm{Adv}^{\text{CKS-light}}_{\mathcal{B},\mathsf{NIKE}}(k, q_C, q'_{CR}) &= \left|\Pr[\mathcal{B}(G'_1) = 1] - \Pr[\mathcal{B}(G'_0) = 1]\right| \\
&\geq (2/q_T q_H^2) \left| \sum_{\iota=0}^{q_T-1} \Pr[\mathcal{A}(G_{\iota+1}) = 1] - \sum_{\iota=0}^{q_T-1} \Pr[\mathcal{A}(G_\iota) = 1] \right| \\
&= (2/q_T q_H^2) \left|\Pr[\mathcal{A}(G_{q_T}) = 1] - \Pr[\mathcal{A}(G_0) = 1]\right| \\
&= (2/q_T q_H^2) \mathrm{Adv}^{\text{m-CKS-heavy}}_{\mathcal{A},\mathsf{NIKE}}(k, q_H, q_C, q_E, q_{HR}, q_{CR}, q_T).
\end{aligned}$$

This concludes our proof. □

Security models in the honest key registration (HKR) setting: For completeness we also provide NIKE security models in the honest key registration setting where dishonest key registration queries are disallowed. An overview of the models is given in Table 3.2. We remark that Theorem 3.1 carries over to the HKR setting simply by setting q_C and q_{CR} to zero in the theorem statement and proof. So all the security models in the HKR setting are also equivalent to one another. As pointed out in the introduction, constructing NIKE schemes in the HKR setting is much easier than in the more realistic DKR setting.

$\mathsf{NIKE.Setup}(1^k)$

$\mathcal{PG}_2 \leftarrow \mathcal{G}_2(1^k)$, where $\mathcal{PG}_2 = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, p, e, \psi)$

$(u_0, u_1, u_2, S) \leftarrow \mathbb{G}_1$, $(hk, ck) \leftarrow \mathsf{Cham.KeyGen}(1^k)$

$params \leftarrow (\mathcal{PG}_2, u_0, u_1, u_2, S, hk)$

Return $params$

$\mathsf{NIKE.KeyGen}(params, id)$

$x \leftarrow \mathbb{Z}_p$, $r \leftarrow \mathcal{R}_{\mathrm{Cham}}$, $Z \leftarrow g_2^x$, $t \leftarrow \mathsf{ChamH}_{hk}(Z||id, r)$

$Y \leftarrow u_0 u_1^t {u_2}^{t^2}$, $X \leftarrow Y^x$

$pk \leftarrow (X, Z, r)$, $sk \leftarrow x$

Return (pk, sk)

$\mathsf{NIKE.SharedKey}(id_1, pk_1, id_2, sk_2)$

If $id_1 = id_2$ return $\perp$

Parse pk_1 as (X_1, Z_1, r_1) and sk_2 as x_2

$t_1 \leftarrow \mathsf{ChamH}_{hk}(Z_1||id_1, r_1)$

If $e(X_1, g_2) \neq e(u_0 u_1^{t_1} {u_2}^{{t_1}^2}, Z_1)$

then $K_{id_1,id_2} \leftarrow \perp$

else $K_{id_1,id_2} \leftarrow e(S^{x_2}, Z_1)$

Return K_{id_1,id_2}

Figure 3.1: The NIKE scheme $\mathsf{NIKE}_{\mathsf{DBDH\text{-}2}}$.

3.3 Constructions for NIKE

3.3.1 A Construction in the Standard Model from Pairings

We specify how to build a NIKE scheme, $\mathsf{NIKE}_{\mathsf{DBDH\text{-}2}}$, that is secure in the CKS-light security model under the DBDH-2 assumption in the standard model. Our construction makes use of a tuple $\mathcal{PG}_2 = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, p, e, \psi)$, output by a parameter generator $\mathcal{G}_2$, and a chameleon hash function $\mathsf{ChamH} : \{0,1\}^* \times \mathcal{R}_{\mathrm{Cham}} \rightarrow \mathbb{Z}_p^*$. This can be instantiated efficiently using the discrete-log based construction from [102] (see Section 2.3.1 for further details of chameleon hash functions). The component algorithms of the scheme $\mathsf{NIKE}_{\mathsf{DBDH\text{-}2}}$ are defined in Figure 3.1.

The check in the NIKE.SharedKey algorithm for valid public keys can be implemented by evaluating the bilinear map twice. It is clear that NIKE.SharedKey defined in this

way satisfies the requirement that entities id_1 and id_2 are able to compute a common key. To see this, note that $e(S^{x_2}, Z_1) = e(S, g_2)^{x_1,x_2}$. The identity space for this construction, $\mathcal{ID}$, is $\{0,1\}^*$, while the space of shared keys is $\mathcal{SHK} = \mathbb{G}_T$. Public keys and parameters are compact. For example, at the 128-bit security level, using BN curves [14] and point compression, public keys consist of 768 bits plus an element from $\mathcal{R}_{\text{Cham}}$.

As stated before, we can prove $\mathsf{NIKE}_{\mathsf{DBDH\text{-}2}}$ to be secure under the DBDH-2 assumption in the sense of the CKS-light security model. Interestingly, our scheme can be generalised to use any *weak* $(2, \text{poly})$-PHF [89] in combination with a chameleon hash function. That is, Y (in the $\mathsf{NIKE.KeyGen}$ algorithm) would be the output of the *weak* $(2, \text{poly})$-PHF on input t, where t is the output of the chameleon hash function. We have given a specific construction here because suitable weak PHFs are currently rare. A further generalisation of our scheme could use any randomised $(2, \text{poly})$-PHF and avoid the chameleon hash, but no constructions for these are currently known.

Theorem 3.2. *Assume* ChamH *is a family of chameleon hash functions. Then the NIKE scheme* $\mathsf{NIKE}_{\mathsf{DBDH\text{-}2}}$ *is secure under the DBDH-2 assumption relative to generator* $\mathcal{G}_2$*. In particular, suppose* $\mathcal{A}$ *is an adversary against* $\mathsf{NIKE}_{\mathsf{DBDH\text{-}2}}$ *in the* CKS-light *security model. Then there exists a DBDH-2 adversary* $\mathcal{B}$ *with:*

$$\text{Adv}^{\text{DBDH-2}}_{\mathcal{B},\mathcal{G}_2}(k) \geq \text{Adv}^{\text{CKS-light}}_{\mathcal{A},\mathsf{NIKE}_{\mathsf{DBDH\text{-}2}}}(k) - 2\,\text{Adv}^{\text{coll}}_{\mathcal{A}_{\mathcal{CH}},\mathsf{ChamH}}(k).$$

Proof. We proceed via a sequence of games. Let S_ι be the event that $\mathcal{A}$ is successful in Game ι.

Game G_0: This is the original attack game as described in the CKS-light security model. By definition, we have that:

$$\text{Adv}^{\text{CKS-light}}_{\mathcal{A},\mathsf{NIKE}_{\mathsf{DBDH\text{-}2}}}(k) = 2\,|\Pr[S_0] - 1/2|.$$

Game G_1: Eliminate hash collisions. In this game, the challenger changes its answers to `register corrupt user` queries as follows: let id_1^* and id_2^* be the identities of the two honest users, and let their public keys be $pk_1^* = (X_1^*, Z_1^*, r_1^*)$, $pk_2^* = (X_2^*, Z_2^*, r_2^*)$, respectively. Let id be the identity of a user that is the subject of

a `register corrupt user` query with $pk = (X, Z, r)$. If $t = \mathsf{ChamH}_{hk}(Z||id, r) = \mathsf{ChamH}_{hk}(Z_1^*||id_1^*, r_1^*)$ or $t = \mathsf{ChamH}_{hk}(Z||id, r) = \mathsf{ChamH}_{hk}(Z_2^*||id_2^*, r_2^*)$, the challenger aborts (note that in this case we have a collision in ChamH_{hk}). Otherwise, it continues as in the previous game.

Let $\mathtt{abort}_{\mathtt{ChamH}}$ be the event that a collision was found. Games G_0 and G_1 are identical unless $\mathtt{abort}_{\mathtt{ChamH}}$ occurs. By the *difference lemma* [131], we have

$$|\Pr[S_1] - \Pr[S_0]| \leq \Pr[\mathtt{abort}_{\mathtt{ChamH}}].$$

Furthermore,

$$\Pr[\mathtt{abort}_{\mathtt{ChamH}}] \leq \mathrm{Adv}^{\mathrm{coll}}_{\mathcal{A}_{\mathcal{CH}},\mathsf{ChamH}}(k).$$

Game G_2: In this game a DBDH-2 adversary $\mathcal{B}$ on inputs $(\mathcal{PG}_2, g_2^a, g_2^b, g_1^c, T)$ (with g_2^a, g_2^b, g_1^c, $T \in \mathbb{G}_2^2 \times \mathbb{G}_1 \times \mathbb{G}_T$), where $a, b, c \in \mathbb{Z}_p$, runs adversary $\mathcal{A}$ against $\mathsf{NIKE}_{\mathsf{DBDH\text{-}2}}$ simulating the challenger's behaviour as in game G_1. $\mathcal{B}$'s job is to determine whether T equals $e(g_1, g_2)^{abc}$ or a random element from $\mathbb{G}_T$, where g_2 is a generator of $\mathbb{G}_2$ and $g_1 = \psi(g_2)$ is a generator of $\mathbb{G}_1$.

$\mathcal{B}$ runs $\mathsf{Cham.KeyGen}(1^k)$ to obtain a key pair for a chameleon hash function, (hk, ck) (here ck is the trapdoor information for the chameleon hash). It then selects $m_1, m_2 \leftarrow \{0,1\}^*$ and $r_1, r_2 \leftarrow \mathcal{R}_{\mathrm{Cham}}$, where $\mathcal{R}_{\mathrm{Cham}}$ is the chameleon hash function's randomness space. $\mathcal{B}$ sets $t_1^* := \mathsf{ChamH}_{hk}(m_1, r_1)$ and $t_2^* := \mathsf{ChamH}_{hk}(m_2, r_2)$.

Let $p(t) = p_0 + p_1 t + p_2 t^2$ be a polynomial of degree 2 over $\mathbb{Z}_p$ such that $p(t_1^*) = p(t_2^*) = 0$. Let $q(t) = q_0 + q_1 t + q_2 t^2$ be a random polynomial of degree 2 over $\mathbb{Z}_p$. Then $\mathcal{B}$ sets $u_i = (g_1^c)^{p_i} g_1^{q_i}$ $(i \in \{0,1,2\})$ and $S = g_1^c$. Since $q_i \leftarrow \mathbb{Z}_p$, we have $u_i \leftarrow \mathbb{G}_1$. Note that then $u_0 u_1^t u_2^{t^2} = (g_1^c)^{p(t)} {g_1}^{q(t)}$. In particular, $Y_1^* = g_1^{q(t_1^*)}$ and $Y_2^* = g_1^{q(t_2^*)}$, where $q(t_1^*)$ and $q(t_2^*)$ are known values.

$\mathcal{B}$ then gives $params = (\mathcal{PG}_2, u_0, u_1, u_2, S, hk)$ to $\mathcal{A}$ and answers the following queries:

- `register honest user`: When $\mathcal{B}$ receives a `register honest user` query for identity id_1^* from adversary $\mathcal{A}$, it uses the trapdoor information ck of the chameleon hash function to obtain $r_1^* \in \mathcal{R}_{\mathrm{Cham}}$ such that $\mathsf{ChamH}_{hk}(g_2^a||id_1^*, r_1^*) = \mathsf{ChamH}_{hk}(m_1, r_1) = t_1^*$. Notice that, according to the definition of chameleon

hash functions (see Section 2.3.1), r_1^* is uniformly distributed over $\mathcal{R}_{\text{Cham}}$ and independent from r_1. Similarly, when $\mathcal{B}$ receives a second `register honest user` query for identity id_2^* from $\mathcal{A}$, it obtains $r_2^* \in \mathcal{R}_{\text{Cham}}$ such that $\mathsf{ChamH}_{hk}(g_2^b||id_2^*, r_2^*) = \mathsf{ChamH}_{hk}(m_2, r_2) = t_2^*$. Then r_2^* is also uniformly distributed over $\mathcal{R}_{\text{Cham}}$. Now $\mathcal{B}$ sets:

$$pk_1^* = ((\psi(g_2^a)^{q(t_1^*)}, g_2^a, r_1^*) \text{ and } pk_2^* = ((\psi(g_2^b)^{q(t_2^*)}, g_2^b, r_2^*).$$

These are correct public keys since $p(t_1^*) = p(t_2^*) = 0$. Note that implicitly $sk_1^* = a$ and $sk_2^* = b$.

- `register corrupt user`: Here, $\mathcal{B}$ receives a public key pk and a string id from $\mathcal{A}$, and registers them. As in the original attack game, $\mathcal{B}$ aborts if id equals one of the honest identities, id_1^* or id_2^*.

- `corrupt reveal`: In order to output the shared key between one of the two honest users, say id_1^*, and a corrupt user, say id, $\mathcal{B}$ first checks if $pk = (X, Z, r)$ is a valid public key using the pairing. If not, it rejects the query. This makes sure that pk is of the form (Y^d, g_2^d, r) for some $d \in \mathbb{Z}_p$, where $Y = (g_1^c)^{p(t)} g_1^{q(t)}$, for $t = \mathsf{ChamH}_{hk}(Z||id, r)$, and $r \in \mathcal{R}_{\text{Cham}}$. This means that $X = (g_1^{cd})^{p(t)} g_1^{dq(t)}$. Thus, g_1^{cd} can be computed from X and $Z = g_2^d$ by:

$$g_1^{cd} = (X/\psi(Z)^{q(t)})^{1/p(t) \bmod p},$$

where we use the property that $p(t) \neq 0 \bmod p$, which follows from the facts that p is a polynomial of degree 2 with roots t_1^*, t_2^* and that $t \neq t_1^*, t_2^*$ (because we have eliminated hash collisions already in Game G_1).

Now writing $pk_1^* = (X_1^*, Z_1^*, r_1^*)$ for the public key of the honest user id_1^*, the shared key between id_1^* and id can be correctly computed as:

$$K_{id_1^*, id} = e(g_1^{cd}, Z_1^*).$$

- `test`: Return T.

This completes our description of $\mathcal{B}$'s simulation. Note that distinguishing the real case from the random case for $\mathcal{A}$ in Game G_2 is equivalent to solving the DBDH-2 problem. To see this, note that for user id_1^*, we have $Z_1^* = g_2^a$ and $X_1^* = \psi(Z_1^*)^{q(t_1^*)}$, while for user id_2^*, we have $Z_2^* = g_2^b$ and $X_2^* = \psi(Z_2^*)^{q(t_2^*)}$. Hence $K_{id_1^*, id_2^*} = e((g_1^c)^b, Z_1^*) = e((g_1^c)^a, Z_2^*) = e(g_1, g_2)^{abc}$.

Now, since $\mathcal{B}$'s simulation properly handles all of $\mathcal{A}$'s queries and sets up all values with the correct distributions, we have: $\Pr[S_2] = \Pr[S_1]$. Let S_B be the event that $\mathcal{B}$ is successful in outputting the correct bit in the DBDH-2 experiment of Definition 2.23. If $\mathcal{B}$ outputs the same bit as $\mathcal{A}$, we have that $\Pr[S_B] = \Pr[S_2]$.

By collecting the probabilities relating the different games, we have

$$\begin{aligned}
\mathrm{Adv}^{\text{CKS-light}}_{\mathcal{A},\mathsf{NIKE}_{\mathsf{DBDH\text{-}2}}} &= 2\,|\Pr[S_0] - 1/2| \\
&\leq 2\,|\Pr[S_1] + \mathrm{Adv}^{\text{coll}}_{\mathcal{A}_{\mathcal{CH}},\mathsf{ChamH}}(k) - 1/2| \\
&\leq 2\,|\Pr[S_2] + \mathrm{Adv}^{\text{coll}}_{\mathcal{A}_{\mathcal{CH}},\mathsf{ChamH}}(k) - 1/2| \\
&= 2\,|\Pr[S_B] + \mathrm{Adv}^{\text{coll}}_{\mathcal{A}_{\mathcal{CH}},\mathsf{ChamH}}(k) - 1/2|.
\end{aligned}$$

Thus,

$$\mathrm{Adv}^{\text{DBDH-2}}_{\mathcal{B},\mathcal{G}_2}(k) = 2\,|\Pr[S_B] - 1/2| \geq \mathrm{Adv}^{\text{CKS-light}}_{\mathcal{A},\mathsf{NIKE}_{\mathsf{DBDH\text{-}2}}}(k) - 2\,\mathrm{Adv}^{\text{coll}}_{\mathcal{A}_{\mathcal{CH}},\mathsf{ChamH}}(k).$$

This concludes our proof. □

Remark: We note that the map ψ in $\mathcal{PG}_2$ is only used in the security proof for the NIKE scheme $\mathsf{NIKE}_{\mathsf{DBDH\text{-}2}}$ and not in the scheme itself.

3.3.2 A Construction in the Random Oracle Model from Factoring

Let $n(k)$ be a function and δ a constant with $0 \leq \delta < 1/2$. Let RSAgen be an algorithm with input 1^k that generates elements (N, P, Q) such that $N = PQ$ is an n-bit Blum integer and all prime factors of $\phi(N)/4$ are pairwise distinct and have at least δn bits. We specify how to build a NIKE scheme, $\mathsf{NIKE}_{\mathsf{fac}}$, that is secure in the CKS-light security model under the factoring assumption relative to RSAgen in the ROM. Our scheme makes use of a hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^k$ which is modelled as a random oracle in the security proof. We assume that identities *id* come from a space with a natural ordering $<$. The component algorithms of the scheme $\mathsf{NIKE}_{\mathsf{fac}}$ are defined in Figure 3.2.

Theorem 3.3. *The scheme* $\mathsf{NIKE}_{\mathsf{fac}}$ *is secure in the ROM under the factoring assumption relative to* RSAgen. *In particular, suppose* $\mathcal{A}$ *is an adversary against*

$\mathsf{NIKE.Setup}(1^k)$

$(N, P, Q) \leftarrow \mathsf{RSAgen}(1^k)$, $g \leftarrow \mathbb{QR}_N^+$, where $\langle g \rangle = \mathbb{QR}_N^+$

$params \leftarrow (\mathsf{H}, N, g)$

Return $params$

$\mathsf{NIKE.KeyGen}(params, id)$

$x \leftarrow \mathbb{Z}_{\lfloor N/4 \rfloor}$, $X \leftarrow g^x$

$pk \leftarrow X$, $sk \leftarrow x$

Return (pk, sk)

$\mathsf{NIKE.SharedKey}(id_1, pk_1, id_2, sk_2)$

If $(id_1 = id_2)$ or $pk_1 \notin \mathbb{QR}_N^+$ or $pk_2 \notin \mathbb{QR}_N^+$ return $\perp$

else if $\begin{cases} id_1 < id_2 \text{ return } \mathsf{H}(id_1, id_2, {pk_1}^{sk_2}) \\ id_2 < id_1 \text{ return } \mathsf{H}(id_2, id_1, {pk_1}^{sk_2}) \end{cases}$

Figure 3.2: The NIKE scheme $\mathsf{NIKE}_{\mathsf{fac}}$.

$\mathsf{NIKE}_{\mathsf{fac}}$ *in the* CKS-light *security model. Then there exists a factoring adversary* $\mathcal{C}$ *with:*

$$\mathrm{Adv}^{\mathrm{fac}}_{\mathcal{C},\mathsf{RSAgen}}(k) \geq \mathrm{Adv}^{\text{CKS-light}}_{\mathcal{A},\mathsf{NIKE}_{\mathsf{fac}}}(k) - \mathcal{O}(2^{-\delta n(k)}).$$

Proof. Suppose $\mathcal{A}$ is an adversary against $\mathsf{NIKE}_{\mathsf{fac}}$ in the CKS-light security model. We first show how to construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to solve the Double Strong Diffie-Hellman (DSDH) problem (see the DSDH assumption in Section 2.1.5) in the group of *Signed Quadratic Residues* ($\mathbb{QR}_N^+$), where N is generated by RSAgen, and then use Theorem 2.7 (Breaking DSDH $\Rightarrow$ Factoring) to construct a factoring adversary $\mathcal{C}$. $\mathcal{B}$'s input is $(N, g, X = g^x, Y = g^y)$, where g is a generator of $\mathbb{QR}_N^+$ and (g^x, g^y) is an instance of the CDH problem in $\mathbb{QR}_N^+$. $\mathcal{B}$'s task is to compute g^{xy}, given access to two decisional oracles $\mathrm{DDH}_{g,X}(\cdot,\cdot)$ and $\mathrm{DDH}_{g,Y}(\cdot,\cdot)$. $\mathcal{B}$ acts as a challenger for $\mathcal{A}$.

$\mathcal{B}$ gives $\mathcal{A}$ the tuple (H, N, g), where H is a random oracle controlled by $\mathcal{B}$. $\mathcal{B}$ maintains a list L, initially empty, to store random oracle responses or responses to shared keys. $\mathcal{A}$ makes a series of queries which $\mathcal{B}$ answers as follows.

- `register honest user`: When $\mathcal{B}$ receives `register honest user` queries for identities id_1^* and id_2^*, $\mathcal{B}$ sets $pk_1^* = X$ and $pk_2^* = Y$.

- `register corrupt user`: Here, $\mathcal{B}$ receives a public key pk and a string id from $\mathcal{A}$, and registers them. As in the original attack game, $\mathcal{B}$ aborts if id equals one of the honest identities, id_1^* or id_2^*.

- `corrupt reveal`: In order to output the shared key between one of the two honest users, say id_1^*, and a corrupt user, say id, $\mathcal{B}$ checks if id_1^* and id already appears in an entry of the form (id_1^*, id, h, R) on the list L, for correct h (without loss of generality, assume $id_1^* < id$). If so, then $\mathcal{B}$ returns $K_{id_1^*,id} = R$ in response to $\mathcal{A}$'s query. Otherwise, $\mathcal{B}$ replies with $R \leftarrow \{0,1\}^k$ and adds $(id_1^*, id, \perp, R)$ to L. Notice that by setup $sk_1^* = x = \mathrm{dlog}_g X$ (unknown to $\mathcal{B}$), so the 'correct' key would be $K_{id_1^*,id} = \mathsf{H}(id_1^*, id, pk_{id}^x)$. (We assume $pk_{id} \in \mathbb{QR}_N^+$, otherwise $\mathcal{B}$ returns $\perp$.)

- `test`: At some point during the simulation, $\mathcal{A}$ makes a single `test` query on the pair of identities (id_1^*, id_2^*). $\mathcal{B}$ outputs a randomly generated value $R \in \{0,1\}^k$. Notice that the 'correct' key $K_{id_1^*,id_2^*}$ that would be computed by $\mathcal{B}$ in responding to this query is equal to $\mathsf{H}(id_1^*, id_2^*, g^{xy})$ (w.l.o.g. assuming $id_1^* < id_2^*$).

- H queries: On input (id_1, id_2, h), w.l.o.g. $id_1, id_2 \in \{0,1\}^*$, $id_1 < id_2$, and $h \in \mathbb{QR}_N^+$, $\mathcal{B}$ answers $\mathcal{A}$'s H queries as follows. $\mathcal{B}$ checks if (id_1, id_2, h, R) is already on the list for some R; If so, it outputs R. Otherwise, $\mathcal{B}$ checks if an entry of the form $(id_1, id_2, \perp, R)$ is on the list for id_1 or id_2 equals one of the two honest identities id_1^* or id_2^* (say $id_1 = id_1^*$ (resp. $id_1 = id_2^*$), that is, $pk_1 = X$ (resp. $pk_1 = Y$)). If so, $\mathcal{B}$ checks if h is the 'correct' DH value for (id_1, id_2) using one of the DDH oracles. (Note that $\mathcal{B}$ does not know the private key $x = \mathrm{dlog}_g X$ (resp. $y = \mathrm{dlog}_g Y$).) That is, if $\mathrm{DDH}_{g,X}(pk_2, h) = 1$ (resp. $\mathrm{DDH}_{g,Y}(pk_2, h) = 1$), then $h = pk_2^x$ (resp. $h = pk_2^y$) and $\mathcal{B}$ updates the entry $(id_1, id_2, \perp, R)$ with (id_1, id_2, h, R) to the list and returns R. Note that entries of the form $(id_1, id_2, \perp, R)$ are only added to the list when one of the strings, id_1 or id_2, is id_1^* or id_2^* and the other is registered. Finally, if no entry of either of the above forms already appears on the list, that is, no (id_1, id_2, h, R) and no $(id_1, id_2, \perp, R)$, then $\mathcal{B}$ picks a random value $R \in \{0,1\}^k$, returns R and adds (id_1, id_2, h, R) to L.

This completes our description of $\mathcal{B}$'s simulation. If $\mathcal{A}$ queries the random oracle H with (id_1^*, id_2^*, h) or (id_2^*, id_1^*, h) for $h = g^{xy}$, then it efficiently solved $\mathcal{B}$'s own

CDH challenge. This can be noticed by $\mathcal{B}$ (with the help of oracle $\mathrm{DDH}_{g,X}(\cdot,\cdot)$ or $\mathrm{DDH}_{g,Y}(\cdot,\cdot)$) and $\mathcal{B}$ can return g^{xy} to its challenger. Note that if $\mathcal{A}$ does not query H on (id_1^*, id_2^*, g^{xy}) or (id_2^*, id_1^*, g^{xy}), then $\mathcal{A}$'s advantage is zero because it cannot distinguish real from random answers to `test` queries. Hence, if $\mathcal{A}$ has non-negligible success probability, then this query must be made by $\mathcal{A}$. We then see that $\mathrm{Adv}^{\mathrm{DSDH}}_{\mathcal{B},\mathsf{RSAgen}}(k) \geq \mathrm{Adv}^{\text{CKS-light}}_{\mathcal{A},\mathsf{NIKE}_{\mathsf{fac}}}(k) - \mathcal{O}(2^{-\delta n(k)})$, where the $\mathcal{O}(2^{-\delta n(k)})$ term accounts for the statistical difference between the distribution of g^x and g^y in the real game (where $x, y \in \mathbb{Z}_{\lfloor N/4 \rfloor}$) and the simulation (where $x, y \in \mathbb{Z}_{\phi(N)/4}$). Combining these facts with Theorem 2.7, we have that $\mathrm{Adv}^{\mathrm{fac}}_{\mathcal{C},\mathsf{RSAgen}}(k) \geq \mathrm{Adv}^{\text{CKS-light}}_{\mathcal{A},\mathsf{NIKE}_{\mathsf{fac}}}(k) - \mathcal{O}(2^{-\delta n(k)})$, concluding the proof. □

3.4 From Non-Interactive Key Exchange to Public Key Encryption

In this section we give a conversion that takes a NIKE scheme that is secure in the CKS-light security model plus a strongly one-time secure signature scheme (strong-OTS), and produces a KEM that is IND-CCA secure. From such a KEM, it is easy to construct an IND-CCA secure public key encryption scheme [46]. (We refer to Secions 2.3.5 and 2.3.6 for formal definitions of an IND-CCA KEM and a strong-OTS scheme.) Let $\mathsf{NIKE} = (\mathsf{NIKE.Setup}, \mathsf{NIKE.KeyGen}, \mathsf{NIKE.SharedKey})$ be a NIKE scheme and $\mathsf{OTS} = (\mathsf{OTS.KG}, \mathsf{OTS.Sign}, \mathsf{OTS.Vfy})$ be a signature scheme. The components of our KEM, $\mathsf{KEM}(\mathsf{NIKE}, \mathsf{OTS})$, are defined in Figure 3.3.

Notice that the ciphertexts in this scheme consist of a verification key from the strong-OTS scheme, a public key from the NIKE scheme, and a one-time signature, while the encapsulated keys are elements of the shared key space, $\mathcal{SHK}$, of the NIKE scheme.

Theorem 3.4. *Suppose the NIKE scheme* NIKE *is secure in the* CKS-light *security model and* OTS *is a strongly one-time secure signature scheme. Then* $\mathsf{KEM}(\mathsf{NIKE}, \mathsf{OTS})$ *is an IND-CCA secure KEM. More precisely, for any adversary* $\mathcal{A}$ *against the IND-CCA security of* $\mathsf{KEM}(\mathsf{NIKE}, \mathsf{OTS})$*, there exists an adversary* $\mathcal{B}$ *against* NIKE *in the* CKS-light *security model or an adversary* $\mathcal{C}$ *against* OTS *having the same advantage as* $\mathcal{A}$*. Moreover, if* $\mathcal{A}$ *makes* q_D *decapsulation queries,*

$\mathsf{KEM.KG}(1^k)$
$params \leftarrow \mathsf{NIKE.Setup}(1^k)$
$id \leftarrow \mathcal{ID}$, $(pk, sk) \leftarrow \mathsf{NIKE.KeyGen}(params, id)$
$pk_{\mathsf{KEM}} := (params, id, pk)$, $sk_{\mathsf{KEM}} := (id, sk)$
Return $(pk_{\mathsf{KEM}}, sk_{\mathsf{KEM}})$

$\mathsf{KEM.Enc}(pk_{\mathsf{KEM}})$
Parse pk_{KEM} as $(params, id, pk)$
Repeat: $(vk, sigk) \leftarrow \mathsf{OTS.KG}(1^k)$, until $vk \neq id$
$(pk', sk') \leftarrow \mathsf{NIKE.KeyGen}(params, id' = vk)$
$\sigma \leftarrow \mathsf{OTS.Sign}(sigk, pk')$
$K \leftarrow \mathsf{NIKE.SharedKey}(id, pk, id' = vk, sk')$
Return $(K, C = (vk, pk', \sigma))$

$\mathsf{KEM.Dec}(sk_{\mathsf{KEM}}, C)$
Parse C as (vk, pk', σ) and sk_{KEM} as (id, sk)
$V \leftarrow \mathsf{OTS.Vfy}(vk, pk', \sigma)$
If $(vk = id)$ or $Y = \texttt{reject}$ return $\perp$, else
Return $\mathsf{NIKE.SharedKey}(id' = vk, pk', id, sk)$

Figure 3.3: The KEM $\mathsf{KEM}(\mathsf{NIKE}, \mathsf{OTS})$.

Note. We assume that *params* contains the security parameter 1^k.

then $\mathcal{B}$ makes at most q_D `register corrupt user` *queries and at most q_D* `corrupt reveal` *queries, while $\mathcal{B}$'s running time is roughly the same as that of $\mathcal{A}$.*

Proof. Let $\mathcal{A}$ be an adversary against the IND-CCA security of $\mathsf{KEM}(\mathsf{NIKE}, \mathsf{OTS})$ (see Definition 2.33). We build $\mathcal{B}$, an adversary against the NIKE scheme in the CKS-light security model (or $\mathcal{C}$, an adversary against the strong one-time security of OTS).

$\mathcal{B}$, on input *params*, a set of system parameters, picks one identity id_1 uniformly at random from the identity space of NIKE, $\mathcal{ID}$, and runs $\mathsf{OTS.KG}(1^k)$ to obtain $(vk, sigk)$. It sets $id_2 := vk$ and makes two `register honest user` queries on id_1 and id_2 receiving public keys pk_1, pk_2. $\mathcal{B}$ then sets $pk_{\mathsf{KEM}} := (params, id_1, pk_1)$. $\mathcal{B}$ also makes a `test` query on id_1, id_2. It receives in reply a value $\hat{K}$, which is either the real key, $K^* = \mathsf{NIKE.SharedKey}(id_1, pk_1, id_2, sk_2)$, or a random key K from $\mathcal{SHK}$. $\mathcal{B}$ sets $C^* := (id_2, pk_2, \sigma^*)$, where $\sigma^* \leftarrow \mathsf{OTS.Sign}(sigk, pk_2)$ and gives $(pk_{\mathsf{KEM}}, \hat{K}, C^*)$

to $\mathcal{A}$.

$\mathcal{A}$ now makes $\mathsf{KEM.Dec}$ queries which $\mathcal{B}$ handles as follows. For each such query with input C, $\mathcal{B}$ parses C as (id', pk', σ') and checks if $id' \neq id_1$ and if σ' is a valid signature on pk' under the verification key $vk' = id'$. If the signature is not valid or $id' = id_1$, $\mathcal{B}$ outputs $\perp$. Otherwise, if $id' = id_2$ (i.e. $vk = vk'$) and $(pk', \sigma') \neq (pk_2, \sigma^*)$, then we can build another adversary $\mathcal{C}$ against the strong one-time security of OTS. (This is done using the same simulation as above with the difference that sk_1 and sk_2 are known but vk comes from the OTS experiment. The signing oracle is used to generate σ^* for the challenge ciphertext.) If $id' = id_2$ and $(pk', \sigma') = (pk_2, \sigma^*)$, C is not a valid query. Assuming $id' \notin \{id_1, id_2\}$, $\mathcal{B}$ makes a `register corrupt user` query on input (id', pk'). $\mathcal{B}$ then makes a `corrupt reveal` query on (id_1, id') and forwards the result to $\mathcal{A}$.

This completes our description of $\mathcal{B}$'s simulation. $\mathcal{A}$'s view is identical when playing either against $\mathcal{B}$ in this simulation or against its real KEM challenger. Note that in the KEM real game, $\mathsf{KEM.Dec}(sk_{\mathsf{KEM}}, C)$, where $C = (vk' = id', pk', \sigma')$, should return $\mathsf{NIKE.SharedKey}(id', pk', id_1, sk_1)$ or $\perp$ if the signature does not verify or if $id' = id_1$. Note also that in the KEM real game, the challenge query should be answered with either $\mathsf{KEM.Enc}(pk_{\mathsf{KEM}}) = (K^*, C^*)$, where $K^* = \mathsf{NIKE.SharedKey}(id_1, pk_1, id_2, sk_2)$ and $C^* = (id_2, pk_2, \sigma^*)$, or a pair (K, C^*), with K chosen at random from $\mathcal{SHK}$. This is exactly what is done in $\mathcal{B}$'s simulation.

Whenever $\mathcal{A}$ outputs a bit $\hat{b}$, $\mathcal{B}$ outputs the same bit. Then we have that $\mathcal{B}$'s advantage in breaking the NIKE scheme is the same as $\mathcal{A}$'s advantage in breaking the KEM. Counting queries made by $\mathcal{B}$ in response to $\mathcal{A}$'s queries completes the proof. □

Applying the above construction to the pairing-based NIKE scheme from Section 3.3.1 results in an IND-CCA secure KEM with public keys (id, pk) that consist of an identity string, two group elements (one in $\mathbb{G}_1$ and one in $\mathbb{G}_2$), and a random value from the randomness space of the Chameleon hash function. Ciphertexts are slightly longer, containing in addition a signature from the one-time signature scheme.[4]

[4] Arguably, one might also include the public parameters *params* when evaluating the public key size.

In the next section we explain how a simplified notion for NIKE and its security lead to a more efficient KEM that is competitive with the BMW scheme from [35], for example.

3.5 Simplified NIKE and Public Key Encryption

Here we show how simplified versions of the NIKE definition and the CKS-light security model can be used to build an IND-CCA secure KEM. The essential difference from our previous definitions is that we eliminate all identities from the algorithms in a NIKE scheme, and assume that the adversary works only with *distinct public keys*, i.e. the adversary is not allowed to register the same public key more than once. (We stress that this simplified version of the NIKE definition and security model do not capture well how real CAs typically operate.) Thus we show that a simpler notion of NIKE than the one we have considered so far suffices for constructing an IND-CCA PKE.

We begin by giving our simplified NIKE definition and security model, and finally the simplified conversion itself. We then illustrate the conversion using a simplified version of the NIKE scheme from Section 3.3.1.

3.5.1 Simplified NIKE and Simplified CKS-light Security Model

We define a simplified NIKE (S-NIKE) scheme, S-NIKE, as a collection of three algorithms S-NIKE.Setup, S-NIKE.KeyGen and S-NIKE.SharedKey, together with a shared key space $\mathcal{SHK}$.

- S-NIKE.Setup: As for NIKE.
- S-NIKE.KeyGen: On input *params*, this probabilistic algorithm outputs a key pair (pk, sk).
- S-NIKE.SharedKey: On input a public key pk_1 and a private key sk_2 (with corresponding public key pk_2), this algorithm outputs either a shared key

$K_{1,2} \in \mathcal{SHK}$, or a failure symbol $\perp$. This algorithm is assumed to always output $\perp$ if $pk_1 = pk_2$.

For correctness, we require that, for any key pairs (pk_1, sk_1) and (pk_2, sk_2), algorithm S-NIKE.SharedKey satisfies the constraint

$$\mathsf{S\text{-}NIKE.SharedKey}(pk_1, sk_2) = \mathsf{S\text{-}NIKE.SharedKey}(pk_2, sk_1).$$

Our simplified CKS-light (S-CKS-light) security model for S-NIKE is stated in terms of a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. In this game, $\mathcal{C}$ takes as input the security parameter 1^k, runs $params \leftarrow \mathsf{S\text{-}NIKE.Setup}(1^k)$ and gives $\mathcal{A}$ $params$. The challenger takes a random bit b and answers oracle queries for $\mathcal{A}$ until $\mathcal{A}$ outputs a bit $\hat{b}$. The challenger answers the following types of queries for $\mathcal{A}$:

- `register honest user`: The challenger $\mathcal{C}$ obtains a public/private key pair by running $(pk, sk) \leftarrow \mathsf{S\text{-}NIKE.KeyGen}(params)$ and records the tuple $(honest, pk, sk)$. $\mathcal{C}$ returns pk to $\mathcal{A}$. $\mathcal{A}$ is allowed to make at most 2 such queries; we refer to the components in the responses as being *honest* keys.
- `register corrupt user`: In this type of query, $\mathcal{A}$ supplies a public key pk. The challenger $\mathcal{C}$ records the tuple $(corrupt, pk, \perp)$.
- `corrupt reveal`: Here $\mathcal{A}$ supplies a pair of public keys pk_1, pk_2, such that one of the keys was registered as *honest* and the other as *corrupt*. The challenger runs S-NIKE.SharedKey using as input the *corrupt* public key and the private key corresponding to the *honest* public key and returns the result to $\mathcal{A}$.
- `test`: Here $\mathcal{A}$ supplies two *honest* public keys pk_1, pk_2. If $b = 1$, the challenger runs $\mathsf{S\text{-}NIKE.SharedKey}(pk_1, sk_2) = \mathsf{S\text{-}NIKE.SharedKey}(pk_2, sk_1)$ and returns the result to $\mathcal{A}$. If $b = 0$, the challenger generates a random key from $\mathcal{SHK}$, records it for later, and returns that to the adversary. $\mathcal{A}$ makes a single test query.

$\mathcal{A}$'s queries may be made adaptively and are arbitrary in number. We demand that all the public keys involved in the `register corrupt user` queries made by $\mathcal{A}$ are

distinct from one another and also distinct from any *honest* public key received by the time of the `register corrupt user` query. When the adversary finally outputs $\hat{b}$, it wins the game if $\hat{b} = b$. For an adversary $\mathcal{A}$, we define its advantage in this security game as:

$$\mathrm{Adv}^{\text{S-CKS-light}}_{\mathcal{A},\mathsf{S\text{-}NIKE}}(k, q_C, q_{CR}) = 2\,|\Pr[\hat{b} = b] - 1/2|,$$

where q_C, q_{CR} are the numbers of `register corrupt user` queries and `corrupt reveal` queries made by $\mathcal{A}$, respectively. Then we say that an S-NIKE scheme is $(t, \epsilon, q_C, q_{CR})$-secure in the S-CKS-light model for S-NIKE if there is no adversary with advantage at least ϵ that runs in time t and makes at most q_C `register corrupt user` queries and at most q_{CR} `corrupt reveal` queries. Informally, we say that an S-NIKE scheme is S-CKS-light secure if there is no efficient adversary having non-negligible advantage in k, where efficient means that the running time and numbers of queries made by the adversary are bounded by polynomials in k. For simplicity we may also use $\mathrm{Adv}^{\text{S-CKS-light}}_{\mathcal{A},\mathsf{S\text{-}NIKE}}(k)$ to denote the advantage of $\mathcal{A}$, in the S-CKS-light model, against a NIKE scheme.

3.5.2 The Conversion from S-NIKE to KEM

We now present our conversion from an S-NIKE scheme to a KEM. Let $\mathsf{S\text{-}NIKE} = (\mathsf{S\text{-}NIKE.Setup}, \mathsf{S\text{-}NIKE.KeyGen}, \mathsf{S\text{-}NIKE.SharedKey})$ be an S-NIKE scheme. The components of our KEM, KEM $\mathsf{KEM(S\text{-}NIKE)}$, are defined in Figure 3.4.

Notice that the ciphertexts in this scheme just consist of public keys from the S-NIKE scheme, while the encapsulated keys are elements of $\mathcal{SHK}$. As we show in Theorem 3.5, the resulting KEM is automatically IND-CCA secure if the S-NIKE scheme is secure in the S-CKS-light security model.

Theorem 3.5. *Suppose the S-NIKE scheme* $\mathsf{S\text{-}NIKE}$ *is secure in the* S-CKS-light *security model. Then* $\mathsf{KEM(S\text{-}NIKE)}$ *is an IND-CCA secure KEM. More precisely, for any adversary* $\mathcal{A}$ *against* $\mathsf{KEM(S\text{-}NIKE)}$*, there exists an adversary* $\mathcal{B}$ *against* $\mathsf{S\text{-}NIKE}$ *in the* S-CKS-light *security model having the same advantage as* $\mathcal{A}$*. Moreover, if* $\mathcal{A}$ *makes* q_D *decapsulation queries, then* $\mathcal{B}$ *makes at most* q_D `register corrupt user` *queries and at most* q_D `corrupt reveal` *queries, while* $\mathcal{B}$*'s running time is roughly the same as that of* $\mathcal{A}$*.*

$\mathsf{KEM.KG}(1^k)$

$params \leftarrow \mathsf{S\text{-}NIKE.Setup}(1^k)$

$(pk, sk) \leftarrow \mathsf{S\text{-}NIKE.KeyGen}(params)$

Return $(pk_{\mathsf{KEM}} := pk, sk_{\mathsf{KEM}} := sk)$

$\mathsf{KEM.Enc}(pk_{\mathsf{KEM}})$

$(pk', sk') \leftarrow \mathsf{S\text{-}NIKE.KeyGen}(params)$

$K \leftarrow \mathsf{S\text{-}NIKE.SharedKey}(pk_{\mathsf{KEM}}, sk')$

Return $(K, C := pk')$

$\mathsf{KEM.Dec}(sk_{\mathsf{KEM}}, C)$

Return $\mathsf{S\text{-}NIKE.SharedKey}(C := pk', sk_{\mathsf{KEM}})$

Figure 3.4: The KEM $\mathsf{KEM(S\text{-}NIKE)}$.

Proof. Let $\mathcal{A}$ be an adversary against $\mathsf{KEM(S\text{-}NIKE)}$. We build $\mathcal{B}$, an adversary against $\mathsf{S\text{-}NIKE}$ in the S-CKS-light security model.

$\mathcal{B}$, on input *params*, a set of system parameters, makes two `register honest user` queries and receives public keys pk_1, pk_2. $\mathcal{B}$ then sets $pk_{\mathsf{KEM}} := pk_1$. $\mathcal{B}$ also makes a `test` query on pk_1, pk_2. It receives in reply a value $\hat{K}$, which is either the real key, $K^* = \mathsf{S\text{-}NIKE.SharedKey}(pk_1, sk_2)$ (where sk_2 is the private key associated with pk_2), or a random key K from $\mathcal{SHK}$. $\mathcal{B}$ sets $C^* := pk_2$ and gives $(pk_{\mathsf{KEM}}, \hat{K}, C^*)$ to $\mathcal{A}$.

$\mathcal{A}$ now makes $\mathsf{KEM.Dec}$ queries which $\mathcal{B}$ handles as follows. For each such query with input C, $\mathcal{B}$ parses C as pk'. Since $C \neq C^*$, we have $pk' \neq pk_2$. If $pk' = pk_1$, then $\mathcal{B}$ outputs $\perp$, which is consistent with the output of the decapsulation algorithm in the scheme $\mathsf{KEM(S\text{-}NIKE)}$ ($\mathsf{S\text{-}NIKE.SharedKey}(pk', sk_{\mathsf{KEM}})$ always outputs $\perp$ if $pk' = pk_{\mathsf{KEM}}$) . Otherwise, $\mathcal{B}$ makes a `register corrupt user` query on input pk' (here we assume, without loss of generality, that all $\mathsf{KEM.Dec}$ queries made by $\mathcal{A}$ are distinct). $\mathcal{B}$ then makes a `corrupt reveal` query on (pk_1, pk') and returns the result to $\mathcal{A}$.

This completes our description of $\mathcal{B}$'s simulation. $\mathcal{A}$'s view is identical when playing either against $\mathcal{B}$ in this simulation or against its real KEM challenger. Note that in the KEM real game $\mathsf{KEM.Dec}(sk_{\mathsf{KEM}}, C)$, where $C = pk'$, should return the output of $\mathsf{S\text{-}NIKE.SharedKey}(pk', sk_1)$. Note also that in the KEM real game, the

---

$\mathsf{S\text{-}NIKE.Setup}(1^k)$

$\mathcal{PG}_2 = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, p, e, \psi) \leftarrow \mathcal{G}_2(1^k)$

$(u_0, u_1, u_2, S) \leftarrow \mathbb{G}_1, \mathsf{TCR}_f \leftarrow \mathsf{TCR}$

$params \leftarrow (\mathcal{PG}_2, u_0, u_1, u_2, S, \mathsf{TCR}_f)$

Return $params$

$\mathsf{S\text{-}NIKE.KeyGen}(params)$

$x \leftarrow \mathbb{Z}_p, Z \leftarrow g_2^x, t \leftarrow \mathsf{TCR}_f(Z), Y \leftarrow u_0 u_1^t {u_2}^{t^2}, X \leftarrow Y^x$

$pk \leftarrow (X, Z),\ sk \leftarrow x$

Return (pk, sk)

$\mathsf{S\text{-}NIKE.SharedKey}(pk_1, sk_2)$

Parse pk_1 as (X_1, Z_1) and sk_2 as x_2

$t_1 \leftarrow \mathsf{TCR}_f(Z_1)$

If $e(X_1, g_2) \neq e(u_0 u_1^{t_1} {u_2}^{{t_1}^2}, Z_1)$

then $K_{1,2} \leftarrow \perp$

else $K_{1,2} \leftarrow e(S^{x_2}, Z_1)$

Return $K_{1,2}$

Figure 3.5: The S-NIKE scheme $\mathsf{SNIKE}_{\mathsf{DBDH\text{-}2}}$.

challenge query should be answered with either $\mathsf{KEM.Enc}(pk_{\mathsf{KEM}}) = (K^*, C^*)$, where $K^* = \mathsf{NIKE.SharedKey}(pk_1, sk_2)$ and $C^* = pk_2$, or a pair (K, C^*), with K chosen at random from $\mathcal{SHK}$. This is exactly what is done in $\mathcal{B}$'s simulation.

Whenever $\mathcal{A}$ outputs a bit $\hat{b}$, $\mathcal{B}$ outputs the same bit. We have that $\mathcal{B}$'s advantage in breaking the S-NIKE scheme is the same as $\mathcal{A}$'s advantage in breaking the KEM. Counting queries made by $\mathcal{B}$ in response to $\mathcal{A}$'s queries completes the proof. □

3.5.3 Applying the Conversion

In Figure 3.5 we consider a simplified version of the NIKE scheme $\mathsf{NIKE}_{\mathsf{DBDH\text{-}2}}$ of Section 3.3.1, which we denote by $\mathsf{SNIKE}_{\mathsf{DBDH\text{-}2}}$. The scheme makes use of a target collision resistant hash function $\mathsf{TCR}_f : \mathbb{G}_2 \rightarrow \mathbb{Z}_p$ chosen uniformly from a family of target collision resistant hash functions TCR (see Section 2.3 for definition of a TCR), instead of a collision resistant hash function.

Theorem 3.6. *Assume* TCR *is a family of target collision resistant hash functions. Then* $\mathsf{SNIKE}_{\mathsf{DBDH\text{-}2}}$ *is secure under the DBDH-2 assumption relative to generator* $\mathcal{G}_2$*. In particular, suppose* $\mathcal{A}$ *is an adversary against* $\mathsf{SNIKE}_{\mathsf{DBDH\text{-}2}}$ *in the* S-CKS-light *security model. Then there exists a DBDH-2 adversary* $\mathcal{B}$ *with:*

$$\mathrm{Adv}^{\text{DBDH-2}}_{\mathcal{B},\mathcal{G}_2}(k) \geq \mathrm{Adv}^{\text{S-CKS-light}}_{\mathcal{A},\mathsf{SNIKE}_{\mathsf{DBDH\text{-}2}}}(k) - 2\,\mathrm{Adv}^{\text{TCR}}_{\mathcal{A}_{\mathcal{H}},\mathsf{TCR}}(k).$$

Proof. We omit the proof as it is similar to that of Theorem 3.2. □

When applying our conversion of Figure 3.4 for the S-NIKE scheme $\mathsf{SNIKE}_{\mathsf{DBDH\text{-}2}}$, we obtain a KEM that benefits from short public keys and private keys (just the public keys and private keys of $\mathsf{SNIKE}_{\mathsf{DBDH\text{-}2}}$) and short ciphertexts (just the public keys of $\mathsf{SNIKE}_{\mathsf{DBDH\text{-}2}}$). This KEM is comparable to the one proposed in [35]: both schemes have the same size of ciphertext, two group elements, and for both, decapsulation requires a ciphertext check and the evaluation of a bilinear map twice. The encapsulation key consists of only one group element. Also, the public keys and private keys generated by our $\mathsf{KEM.KG}$ algorithm are shorter than the ones in the KEM proposed in [35] (two group elements and one element in $\mathbb{Z}_p$, respectively).

3.6 Chapter Summary

In this chapter we provided different security models for NIKE and explored the relationships between them. We then gave constructions for secure NIKE in the standard model and in the ROM. We also studied the relationship between NIKE and PKE, showing that a secure NIKE implies an IND-CCA secure PKE scheme.

There are several interesting open problems that arise from our work. One is to construct pairing-free NIKE schemes in the standard model. A challenge to doing so is that our pairing-based construction uses the pairing in a fundamental way in order to provide a publicly computable check on the validity of public keys. The RSA/factoring setting seems particularly challenging in this respect.

Another open problem is to construct three-user (or even multi-user) NIKE schemes

based on Joux's protocol [93]. Currently, there is no security model for such schemes, and no constructions which can handle adversarially-generated public keys.

Finally, a very interesting problem is to construct ID-based NIKE schemes that are provably secure in the standard model, moving beyond the ROM schemes analysed in [57, 112]. We will see how to construct such schemes in the next chapter.

Chapter 4

Identity-based Non-Interactive Key Exchange (ID-NIKE)

Contents

In this chapter we construct the first identity-based non-interactive key exchange (ID-NIKE) scheme with security in the standard model. Additionally, we also construct the first fully secure hierarchical ID-NIKE scheme with security either in the random oracle model or in the standard model. Our constructions are based on multilinear maps and use a variant of a PHF, which we call MPHF. Most of the content of this chapter appears in [64], which is joint work with Dennis Hofheinz, Kenneth G. Paterson and Christoph Striecks.

4.1 Introduction

In Chapter 3 we studied non-interactive key exchange in the public key setting, where, using some common parameters, any user can compute a public/private key pair on its own. The public key should be registered with a Certification Authority (CA), who keeps a directory of all registered public keys where users can look up other users' public keys. Then any pair of users who know each other's public keys can generate a shared key without exchanging any information. In *identity-based* non-interactive key exchange (ID-NIKE), an arbitrary string that uniquely identifies a user (such as an e-mail address or an IP address) can serve as the user's public key. We call this string an *identity* or *identifier* and denote it by id. In this setting, the private key corresponding to id cannot be computed by the user itself anymore. A Trusted Authority (TA), who holds a master public key mpk and a master secret key msk, is responsible for generating and distributing private keys to the right users. Now, any pair of users who have each received a private key from the TA and who know each other's identities can compute a shared key without exchanging any information.

ID-NIKE has important applications in managing keys and enabling secure communications in mobile *ad hoc* and sensor networks where the energy cost of communication is at a premium [42, 70, 73]. In the hierarchical setting, H-ID-NIKE allows the same functionality, but also allows the TA's operations to be distributed over a hierarchy, which is well-suited to military and emergency response scenarios. The advantages of ID-NIKE, in terms of reducing communication costs and latency in a realistic adversarial environment, are demonstrated in [42].

However, ID-NIKE has proven surprisingly hard to instantiate in the standard model, even more so in a hierarchical setting. Currently, to the best of our knowledge, there is precisely one efficient, secure ID-NIKE scheme with a proof of security in the random oracle model, namely the SOK scheme [122] (with security models and analyses in [57, 112]). There are no schemes secure in the standard model.

In the hierarchical setting, Gennaro *et al.* [73] constructed H-ID-NIKE schemes that are secure under certain classes of key exposure, but which do not offer *full security*, the desirable and natural generalisation of the existing ID-NIKE security notion

from [112] to the hierarchical setting. Moreover, their schemes do not scale well to large numbers of levels as the amount of secret information held by the TA grows exponentially with the depth of the hierarchy. The same criticisms apply to earlier schemes [29, 115] on which the schemes of Gennaro *et al.* [73] are based. Indeed, one of the open problems left in [73] is to construct an H-ID-NIKE scheme with security against *not only* compromise of any number of leaves, *but also* against any number of nodes at higher levels of the hierarchy.[1]

4.1.1 Our Contributions

This chapter is aimed at constructing the first ID-NIKE scheme in the literature with provable security in the standard model and additionally, the first H-ID-NIKE scheme with full security in either the random oracle model or the standard model. As a warm-up we start by giving a pairing-based construction towards the first ID-NIKE scheme with security in the standard model. However, this scheme does not achieve the desired level of security, as in the security proof the number of oracle queries that an adversary against the scheme can make is upper bounded by a constant q. This happens due to the main components of our scheme (apart from a pairing) being $(q, 1)$-PHFs for fixed q. Having a $(\text{poly}, 1)$-PHF would give us the desired level of security for our ID-NIKE construction. Unfortunately, a recent result [84] shows that no black-box construction of (poly, n)-PHFs, for constant n, exists.

We circumvent the black-box impossibility result of [84] by slightly adapting the PHF definition to a setting with multilinear maps to obtain what we call MPHFs. We construct (poly, n)-MPHFs and show how our (standard model) $(\text{poly}, 2)$-MPHF can replace the random oracle in the SOK ID-NIKE scheme to obtain a standard model version of that scheme. Our scheme is the first ID-NIKE scheme with provable security in the standard model. Our MPHF constructions also allow us to derive a hierarchical version of the SOK scheme in settings with multilinear maps. Our H-ID-NIKE scheme is the first such scheme with full security in either the random oracle model or the standard model. The analyses of our schemes are completely modular: we prove the security of the MPHF-based schemes solely from generic

[1] We note that there are other papers claiming to solve this open problem (*eg.* [81]), but these can be easily shown to provide insecure schemes.

MPHF properties. In particular, we can also view (programmable) random oracles as MPHFs. So we can either instantiate our schemes with random oracles to obtain reasonably efficient schemes, or with our new MPHFs to obtain (somewhat less efficient) standard-model secure schemes. In this chapter, we also show how multilinear maps can be used to achieve security in the broader scenario of multiple TAs, and for shared keys among whole groups of users.

While our constructions are formulated with respect to a generic multilinear map, we also outline the necessary adaptations required for the recent "noisy" multilinear map candidate due to Garg, Gentry, and Halevi [72].

4.2 Preliminaries

4.2.1 The GGH Candidate Multilinear Maps

Recently, Garg, Gentry, and Halevi [72] have announced a candidate for a family of cryptographically interesting multilinear maps, named *graded encoding systems*. Their candidate is lattice-based, heavily relies on the notion of noise, and thus does not provide groups in the usual sense. With the GGH candidate, group elements have a randomized (and thus non-unique) representation dubbed an "encoding". While it is possible to extract a unique "canonical bitstring" from an encoding, it is not possible to perform further computations with this extracted bitstring. An encoding can be re-randomized (e.g. to hide the sequence of operations that were performed), but only at the cost of introducing an artificial "noise" term in the encoding. Further operations (and re-randomizations) on this group element cause the noise to grow; once this noise grows beyond a certain bound, encodings can no longer be worked with.

4.2.2 Our Abstraction of the GGH Candidate

For readability and universality, in this thesis we will generally use the notation from the abstract notion of multilinear maps described in Section 2.1.4. When instantiated with the GGH candidate, operations are meant to occur on encodings, without

implicit re-randomizations. In particular, e.g. g now denotes an encoding (not a group element). Additionally, we will employ the following notation to indicate necessary re-randomizations, extractions, and comparisons when using encodings instead of group elements.

- $g \leftarrow \mathbb{G}_i$ means choosing a random encoding g at level i. (This corresponds to uniformly choosing a group element from $\mathbb{G}_i$.) We assume that encodings g chosen in such a way have a low noise level, say, 1.
- $g \overset{\mathsf{enc}}{=} h$ holds if and only if the encodings g and h match.
- $g \overset{\mathsf{grp}}{=} h$ holds if and only if the *group elements* encoded by g and h match, that is, if and only if the GGH isZero procedure identifies $g^{-1}h$ as the neutral element.[2]
- $\mathsf{reRand}_j(g)$ is the re-randomization of encoding g. This re-randomization increases the noise level to a certain, a-priori fixed bound j. For simplicity, and abstracting, we only consider noise levels $j \in \mathbb{N}$. If g's noise level is already at least j (e.g. because g is the output of reRand_j), then re-randomization fails. We note that the distributions $\mathsf{reRand}_j(g)$ and $\mathsf{reRand}_j(h)$ are statistically close for any two encodings g, h with $g \overset{\mathsf{grp}}{=} h$ and noise level less than j.
- $\mathsf{ext}(g)$ denotes the canonical bitstring extracted from encoding g. We have $\mathsf{ext}(g) = \mathsf{ext}(h)$ for any g, h with $g \overset{\mathsf{grp}}{=} h$ of sufficiently small noise level.

Like [72], we omit parameters (such as noise bounds) to computations; asymptotic parameters can be derived from the suggestions in [72, Section 4.2].

4.3 (Hierarchical) Identity-based Non-Interactive Key Exchange

Hierarchical identity-based non-interactive key exchange (H-ID-NIKE) is the natural generalisation of ID-NIKE [57, 112, 122] to the hierarchical setting: a root authority

[2] Technically, the GGH isZero procedure only allows to compare two encodings on the "highest level" ℓ. To compare two level-i encodings (for $i < \ell$), we can first "lift" both to level ℓ by pairing them with a nonzero level-$(\ell - i)$ element.

calculates and distributes private keys to sub-authorities, who in turn do the same for sub-sub-authorities, and so on, until leaf nodes are reached. Each node is identified by a vector of identities, and any pair of nodes in the tree should be able to non-interactively compute a common key based on their private keys and identities.[3] We recall from the introduction that H-ID-NIKE schemes are rare, and, to the best of our knowledge, there are not even any ROM constructions that meet all the desirable criteria (efficiency, scalability, and full security in the sense of resilience to arbitrary node compromises).

Formally, an H-ID-NIKE scheme $\mathsf{H\text{-}ID\text{-}NIKE}$ consists of three PPT algorithms (see below), an identity space $\mathcal{ID}$ and shared-key space $\mathcal{SHK}$. The users are organized in a tree of depth L whose root (at level 0) is the trusted authority (TA). The identity of a user at level $d \in [L]$ is represented by a vector $\mathbf{id} = (id_1, \ldots, id_d) \in \mathcal{ID}^d$.

Setup. The setup algorithm $\mathsf{Setup}(1^k, L)$ is run by the TA. Given the security parameter 1^k and a parameter $L \in \mathbb{N}$, it outputs a master public key mpk and a master secret key msk. We also interpret msk as the user private key usk_ε for the empty identity ε.

Key delegation. The key delegation algorithm $\mathsf{Del}(mpk, usk_{\mathbf{id}}, \mathbf{id}')$ can be run by any user to generate a private key for any of its children. Given the master public key mpk, the user private key $usk_{\mathbf{id}}$ for an identity $\mathbf{id} = (id_1, \ldots, id_d) \in \mathcal{ID}^d$, the algorithm outputs a user private key $usk_{\mathbf{id}'}$ for any of its children $\mathbf{id}' = (id_1, \ldots, id_d, id_{d+1}) \in \mathcal{ID}^{d+1}$ (for $0 \leq d < L$).

Shared key generation. Given the master public key mpk, a user private key $usk_{\mathbf{id}_1}$ for an identity $\mathbf{id}_1 \in \mathcal{ID}^{\leq L}$, and an identity $\mathbf{id}_2 \in \mathcal{ID}^{\leq L}$, $\mathsf{ShK}(mpk, usk_{\mathbf{id}_1}, \mathbf{id}_2)$ outputs either a shared key $K_{\mathbf{id}_1,\mathbf{id}_2} \in \mathcal{SHK}$ or a failure symbol $\perp$. (If $\mathbf{id}_1$ is an ancestor of $\mathbf{id}_2$ (or vice-versa) the algorithm is assumed to always output $\perp$ [4]; here, $\mathbf{id}$ is in particular considered to be an ancestor of itself. Otherwise the output is assumed to be in $\mathcal{SHK}$.)

For correctness, we require that for any $k, L \in \mathbb{N}$, for any $(mpk, msk) \leftarrow \mathsf{Setup}(1^k, L)$,

[3] We remark that sometimes H-ID-NIKE schemes are more restricted and shared keys can only be computed, for example, between pairs of leaf nodes. This is the case of the schemes of [73].

[4] If $\mathbf{id}_1$ is an ancestor of $\mathbf{id}_2$, it can always compute the user private key $usk_{\mathbf{id}_2}$; a key derived from $usk_{\mathbf{id}_2}$ can be used as a shared key between the two users.

for any pair of identities $(\mathbf{id}_1, \mathbf{id}_2) \in \mathcal{ID}^{d_1} \times \mathcal{ID}^{d_2}$, such that neither is an ancestor of the other, and corresponding user private keys $usk_{\mathbf{id}_1}$ and $usk_{\mathbf{id}_2}$ generated by repeated applications of Del from $usk_\varepsilon = msk$, we have $\mathsf{ShK}(mpk, usk_{\mathbf{id}_1}, \mathbf{id}_2) = \mathsf{ShK}(mpk, usk_{\mathbf{id}_2}, \mathbf{id}_1)$.

A (non-hierarchical) ID-NIKE scheme is an H-ID-NIKE scheme in which the depth L of the tree is fixed to $L = 1$. (Note that in this case, Del gets as input $usk_\varepsilon = msk$ and outputs user private keys for identities at level 1. We may thus also speak of extraction of user private keys.)

4.3.1 Security Definition for (H-)ID-NIKE

We present a security model for H-ID-NIKE that is the natural generalisation of the *PS* (Paterson and Srinivasan) model for ID-NIKE from [112] to the hierarchical setting. The model significantly strengthens the previous model of Gennaro *et al.* [73] by being fully adaptive, allowing arbitrary numbers of node corruptions, and allowing the adversary access to shared keys as well as user private keys of inner (i.e. non-leaf) nodes.

The model is defined in terms of a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. $\mathcal{C}$ takes as input the security parameter 1^k and a depth L, runs algorithm Setup of the H-ID-NIKE scheme and gives $\mathcal{A}$ the master public key mpk. It keeps the master secret key, msk, to itself. $\mathcal{A}$ then makes queries of the following three types:

Extract. $\mathcal{A}$ supplies an identity $\mathbf{id} = (id_1, \ldots, id_d) \in \mathcal{ID}^d$ (for $d \in [L]$). $\mathcal{C}$ uses Del repeatedly, starting from msk, to derive $usk_{\mathbf{id}}$ and hands $usk_{\mathbf{id}}$ to $\mathcal{A}$.

Reveal. Here $\mathcal{A}$ supplies a pair $(\mathbf{id}_1, \mathbf{id}_2) \in \mathcal{ID}^{d_1} \times \mathcal{ID}^{d_2}$. $\mathcal{C}$ extracts $usk_{\mathbf{id}_1}$ as above, runs $K_{\mathbf{id}_1,\mathbf{id}_2} \leftarrow \mathsf{ShK}(mpk, usk_{\mathbf{id}_1}, \mathbf{id}_2)$, and hands $K_{\mathbf{id}_1,\mathbf{id}_2}$ to $\mathcal{A}$.

Test. $\mathcal{A}$ supplies two *challenge* identities $(\mathbf{id}_1^*, \mathbf{id}_2^*) \in \mathcal{ID}^{d_1^*} \times \mathcal{ID}^{d_2^*}$ such that neither is an ancestor of the other. $\mathcal{C}$ computes $K_{\mathbf{id}_1^*,\mathbf{id}_2^*}$ as above, and tosses a coin $b \leftarrow \{0, 1\}$. If $b = 1$ then $\mathcal{C}$ gives $K_{\mathbf{id}_1^*,\mathbf{id}_2^*}$ to $\mathcal{A}$; otherwise, if $b = 0$, then $\mathcal{C}$ gives $\mathcal{A}$ a uniform element from $\mathcal{SHK}$.

Finally, $\mathcal{A}$ outputs a guess $\hat{b}$ for b. In our security model, the adversary is allowed to make an arbitrary (but polynomial) number of ***Extract*** and ***Reveal*** queries, and a single ***Test*** query. Furthermore, the adversary is fully adaptive, in the sense that it can compromise nodes (by making ***Extract*** and/or ***Reveal*** queries) in any order. In order to prevent the adversary from trivially winning, we require that the adversary is not allowed to make any ***Extract*** queries on an ancestor of $\mathbf{id}_1^*$ or $\mathbf{id}_2^*$, and no ***Reveal*** query on the pairs $(\mathbf{id}_1^*, \mathbf{id}_2^*)$ and $(\mathbf{id}_2^*, \mathbf{id}_1^*)$. The advantage of an adversary $\mathcal{A}$ against an H-ID-NIKE scheme $\mathsf{H\text{-}ID\text{-}NIKE}$ is defined as:

$$\mathrm{Adv}^{\text{IND-SK}}_{\mathcal{A},\mathsf{H\text{-}ID\text{-}NIKE}}(k) = 2\left|\Pr[\hat{b} = b] - 1/2\right| = \left|\Pr\left[\hat{b} = 1 \mid b = 1\right] - \Pr\left[\hat{b} = 1 \mid b = 0\right]\right|.$$

We say that a scheme $\mathsf{H\text{-}ID\text{-}NIKE}$ is IND-SK secure if and only if $\mathrm{Adv}^{\text{IND-SK}}_{\mathcal{A},\mathsf{H\text{-}ID\text{-}NIKE}}(k)$ is negligible for all PPT adversaries $\mathcal{A}$.

In the non-hierarchical case (i.e. $L = 1$), we recover the definition and security model for (non-hierarchical) ID-NIKE from [112]. Note also that versions of these models in which multiple ***Test*** queries are permitted for a single bit b can be shown to be polynomially equivalent to the versions with a single ***Test*** query using standard hybrid arguments.

4.4 Towards a Secure ID-NIKE Scheme in the Standard Model

We specify how to obtain an ID-NIKE scheme, $\mathsf{TIDNIKE_{PHF}}$, that is secure in a variant of the *PS* security model under the DBDH assumption in the standard model; we consider a weaker security model where the total number of *non-challenge* identities involved in ***Extract*** and ***Reveal*** queries made by an adversary against the scheme is limited by a constant q (and the usual restrictions on the adversary to prevent trivial wins still apply). We will refer to this security model as the *q-bounded PS* security model. $\mathsf{TIDNIKE_{PHF}}$ is the result of joint work with Dennis Hofheinz.

Our construction is based on the well-known ID-NIKE scheme of Sakai, Ohgishi and Kasahara (SOK) [122]. It makes use of a tuple $\mathcal{PG} = (\mathbb{G}, \mathbb{G}_T, g, p, e)$, output by a pairing parameter generator $\mathcal{G}$ on input a security parameter 1^k, and two instances of a $(q+1, 1)$-PHF $\mathsf{H} = (\mathsf{PHF.Gen}, \mathsf{PHF.Eval})$ with input in $\{0,1\}^k$ and output

Algorithm $\mathsf{Setup}(1^k)$
$\mathcal{PG} = (\mathbb{G}, \mathbb{G}_T, p, g, e) \leftarrow \mathcal{G}(1^k)$
$x \leftarrow \mathbb{Z}_p, \kappa_i \leftarrow \mathsf{PHF.Gen}(1^k)\ (i = 1, 2)$
$mpk := (\mathcal{PG}, \kappa_1, \kappa_2), msk := x$
Return (mpk, msk)

Algorithm $\mathsf{Ext}(mpk, msk, id)$
$Y_{id} \leftarrow \mathsf{H}_{\kappa_1}(id)\mathsf{H}_{\kappa_2}(id)$
$usk_{id} \leftarrow Y_{id}^x$
Return usk_{id}

Algorithm $\mathsf{ShK}(mpk, usk_{id_1}, id_2)$
If $id_1 = id_2$ return $\perp$
$Y_{id_2} \leftarrow \mathsf{H}_{\kappa_1}(id_2)\mathsf{H}_{\kappa_2}(id_2)$
$K_{id_1,id_2} := e(usk_{id_1}, Y_{id_2})$
Return K_{id_1,id_2}

Figure 4.1: The ID-NIKE scheme $\mathsf{TIDNIKE}_{\mathsf{PHF}}$.

in $\mathbb{G}$. The component algorithms of our ID-NIKE scheme $\mathsf{TIDNIKE}_{\mathsf{PHF}}$ are defined in Figure 4.1. (For compatibility with existing notation, we present an extraction algorithm Ext instead of an equivalent delegation algorithm.)

It is clear that our ShK algorithm satisfies the correctness requirement that guarantees that id_1 and id_2 are able to compute a common key. To see this, note that

$$e(usk_{id_1}, Y_{id_2}) = e(Y_{id_1}, Y_{id_2})^x = e(Y_{id_2}, Y_{id_1})^x = e(usk_{id_2}, Y_{id_1}).$$

Theorem 4.1. *Let* H *be a* $(q+1, 1, \gamma, \delta)$*-PHF into* $\mathbb{G}$*. Then* $\mathsf{TIDNIKE}_{\mathsf{PHF}}$ *is secure in the* q-bounded PS *security model under the DBDH assumption relative to generator* $\mathcal{G}$*. In particular, suppose* $\mathcal{A}$ *is an adversary against* $\mathsf{TIDNIKE}_{\mathsf{PHF}}$ *in the* q-bounded PS *security model. Then there exists a DBDH adversary* $\mathcal{B}$ *with*

$$\mathrm{Adv}^{\mathrm{DBDH}}_{\mathcal{B},\mathcal{G}}(k) \geq (\mathrm{Adv}^{\text{IND-SK}}_{\mathcal{A},\mathsf{TIDNIKE}_{\mathsf{PHF}}}(k) + 1 - 4\gamma)\delta - 1.$$

Proof. We proceed with a sequence of games. Let S_ι be the event that $\mathcal{A}$ is successful in Game G_ι.

Game G_0. Let G_0 be the original attack game as described in the *PS* security

model. By definition, we have that

$$\mathrm{Adv}^{\text{IND-SK}}_{\mathcal{A},\mathsf{TIDNIKE}_{\mathsf{PHF}}}(k) = 2\,|\Pr[S_0] - 1/2|\,.$$

Game G_1. In this game, instead of obtainning κ_1, κ_2 from the key generation algorithm $\mathsf{PHF.Gen}$, the challenger runs the trapdoor key generation algorithm twice, $(\kappa'_i, t_i) \leftarrow \mathsf{PHF.TrapGen}(1^k, h_1, h_2)$ $(i = 1, 2)$ for generators h_1 and $h_2 \in \mathbb{G}$, to obtain a pair of keys and trapdoors (κ'_1, t_1), (κ'_2, t_2). Since H is $(q+1, 1, \gamma, \delta)$-programmable, we have

$$|\Pr[S_1] - \Pr[S_0]| \leq 2\gamma.$$

Game G_2. In this game we replace the challenger with a DBDH adversary $\mathcal{B}$. Assume $\mathcal{B}$ gets a tuple $\mathcal{PG} = (\mathbb{G}, \mathbb{G}_T, p, g, e)$ and group elements $(g^a, g^b, g^c, T) \in \mathbb{G}^3 \times \mathbb{G}_T$ as input, where $\langle g \rangle = \mathbb{G}$, $|\mathbb{G}| = |\mathbb{G}_T| = p$ and $a, b, c \in \mathbb{Z}_p$. $\mathcal{B}$'s job is to determine whether T equals $e(g, g)^{abc}$ or a random element from $\mathbb{G}_T$. $\mathcal{B}$ simulates the challenger's behaviour in Game G_1. First it runs $(\kappa'_1, t_1) \leftarrow \mathsf{PHF.TrapGen}(1^k, g^b, g)$, $(\kappa'_2, t_2) \leftarrow \mathsf{PHF.TrapGen}(1^k, g^c, g)$, and then it sets $mpk := (\mathcal{PG}, \kappa'_1, \kappa'_2)$.

During the simulation, $\mathcal{B}$, based on the $(q+1, 1)$-programmability of the PHF H, will hope that for all non-challenge identities id_i $(1 \leq i \leq t \leq q)$ involved in $\mathcal{A}$'s ***Extract*** and ***Reveal*** queries and for challenge identities id^*_1, id^*_2 subject to $\mathcal{A}$'s ***Test*** query, we have that:

$$\alpha_{1,id_1} = \ldots = \alpha_{1,id_t} = 0,\ \alpha_{1,id^*_2} = 0,\ \alpha_{1,id^*_1} \neq 0$$

and

$$\alpha_{2,id_1} = \ldots = \alpha_{2,id_t} = 0,\ \alpha_{2,id^*_1} = 0,\ \alpha_{2,id^*_2} \neq 0.$$

Here, for an identity id, $\alpha_{1,id}$ and $\alpha_{2,id}$ are output by $\mathsf{PHF.TrapEval}(t_1, id)$ and $\mathsf{PHF.TrapEval}(t_2, id)$, respectively. We denote the event that $\mathcal{B}$'s hope is satisfied by good.

In order to answer an ***Extract***(id) query made by $\mathcal{A}$, $\mathcal{B}$ computes $(\alpha_{1,id}, \beta_{1,id}) \leftarrow \mathsf{PHF.TrapEval}(t_1, id)$, $(\alpha_{2,id}, \beta_{2,id}) \leftarrow \mathsf{PHF.TrapEval}(t_2, id)$ and aborts if $\alpha_{1,id} \neq 0$ or $\alpha_{2,id} \neq 0$. Assuming that $\mathcal{B}$ did not abort, we should have

$$Y_{id} = \mathsf{H}_{\kappa'_1}(id)\mathsf{H}_{\kappa'_2}(id) = (g^b)^{\alpha_{1,id}} g^{\beta_{1,id}} (g^c)^{\alpha_{2,id}} g^{\beta_{2,id}} = g^{\beta_{1,id}+\beta_{2,id}}.$$

Using knowledge of g^a, $\mathcal{B}$ can compute $usk_{id} = (g^a)^{\beta_{1,id}+\beta_{2,id}} = Y^a_{id}$. $\mathcal{B}$ gives usk_{id} to $\mathcal{A}$. Note that implicitly, $\mathcal{B}$ is setting $msk := a$.

Additionally, if it is the case that $\alpha_{1,id} = \alpha_{2,id} = 0$ for all non-challenge identities id, then $\mathcal{B}$ can answer not only all ***Extract***, but also all ***Reveal*** queries made by $\mathcal{A}$. For the latter, $\mathcal{B}$ first computes usk_{id} for a non-challenge identity id involved in the ***Reveal*** query, and then computes the shared key.

For the ***Test***(id_1^*, id_2^*) query, if $\alpha_{1,id_1^*}, \alpha_{2,id_2^*} \neq 0$ and $\alpha_{1,id_2^*} = \alpha_{2,id_1^*} = 0$, we have that

$$\begin{aligned}
Y_{id_1^*} &= \mathsf{H}_{\kappa_1'}(id_1^*)\mathsf{H}_{\kappa_2'}(id_1^*) \\
&= (g^b)^{\alpha_{1,id_1^*}} g^{\beta_{1,id_1^*}} (g^c)^{\alpha_{2,id_1^*}} g^{\beta_{2,id_1^*}} \\
&= g^{b\alpha_{1,id_1^*} + (\beta_{1,id_1^*} + \beta_{2,id_1^*})} \\
&= g^{b\alpha_{1,id_1^*} + \beta_{id_1^*}} \\
&\quad \text{and} \\
Y_{id_2^*} &= \mathsf{H}_{\kappa_1'}(id_2^*)\mathsf{H}_{\kappa_2'}(id_2^*) \\
&= (g^b)^{\alpha_{1,id_2^*}} g^{\beta_{1,id_2^*}} (g^c)^{\alpha_{2,id_2^*}} g^{\beta_{2,id_2^*}} \\
&= g^{c\alpha_{2,id_2^*} + (\beta_{1,id_2^*} + \beta_{2,id_2^*})} \\
&= g^{c\alpha_{2,id_2^*} + \beta_{id_2^*}}.
\end{aligned}$$

(To simplify the above equation, for an identity id, we have denoted $\beta_{id} = \beta_{1,id} + \beta_{2,id}$.)

Now, with $msk = a$, the correct shared key between id_1^* and id_2^* should be

$$\begin{aligned}
K_{id_1^*, id_2^*} &= e(Y_{id_1^*}, Y_{id_2^*})^a \\
&= e(g^{b\alpha_{1,id_1^*} + \beta_{id_1^*}}, g^{c\alpha_{2,id_2^*} + \beta_{id_2^*}})^a \\
&= e(g,g)^{a(b\alpha_{1,id_1^*} + \beta_{id_1^*})(c\alpha_{2,id_2^*} + \beta_{id_2^*})} \\
&= e(g,g)^{abc(\alpha_{1,id_1^*}\alpha_{2,id_2^*})} \cdot C,
\end{aligned}$$

where $C = e(g^a, g^b)^{\alpha_{1,id_1^*}\beta_{id_2^*}} e(g^a, g^c)^{\alpha_{2,id_2^*}\beta_{id_1^*}} e(g^a, g)^{\beta_{id_1^*}\beta_{id_2^*}}$ can be computed by $\mathcal{B}$.

$\mathcal{B}$ responds to $\mathcal{A}$'s ***Test*** query with $T^{\alpha_{1,id_1^*}\alpha_{2,id_2^*}} C$. When $T = e(g,g)^{abc}$, this corresponds to the real shared key $K_{id_1^*, id_2^*}$, while when T is random in $\mathbb{G}_T$, $\mathcal{B}$'s response corresponds to a random value in $\mathcal{SHK}$. Thus $\mathcal{B}$'s response to $\mathcal{A}$'s ***Test*** query is properly distributed in Game G_2 (because $T = e(g,g)^{abc}$ or T is random, each with probability 1/2).

$\mathcal{B}$ will abort its simulation at any point if it realizes that $\texttt{good}$ did not occur. Let $\texttt{abort}_{\mathsf{PHF}}$ be the event that $\mathcal{B}$ aborts during simulation. Due to the programmability of H we have that

$$\Pr[S_2] = \Pr[S_1 \wedge \neg \texttt{abort}_{\mathsf{PHF}}] \geq \delta \Pr[\mathtt{S}_1].$$

Finally, $\mathcal{B}$ outputs the same bit that $\mathcal{A}$ outputs. Let S_B be the event that $\mathcal{B}$ is successul in outputting the correct bit in the DBDH experiment of Definition 2.22. It is clear that $\Pr[S_B] = \Pr[S_2]$.

By collecting the probabilities relating the different games, we have

$$\begin{aligned} \mathrm{Adv}^{\text{IND-SK}}_{\mathcal{A},\mathsf{TIDNIKE}_{\mathsf{PHF}}}(k) &= 2|\Pr[S_0] - 1/2| \\ &\leq 2|\Pr[S_1] + 2\gamma - 1/2| \\ &\leq 2|\Pr[S_2]/\delta + 2\gamma - 1/2| \\ &= 2|\Pr[S_B]/\delta + 2\gamma - 1/2|. \end{aligned}$$

Thus,

$$\mathrm{Adv}^{\text{DBDH}}_{\mathcal{B},\mathcal{G}}(k) = 2|\Pr[S_B] - 1/2| \geq (\mathrm{Adv}^{\text{IND-SK}}_{\mathcal{A},\mathsf{TIDNIKE}_{\mathsf{PHF}}}(k) + 1 - 4\gamma)\delta - 1.$$

This concludes our proof. □

Remark: While this would seem to be the first ID-NIKE scheme with security in the standard model, we stress that $\mathsf{TIDNIKE}_{\mathsf{PHF}}$ is not secure in the sense of a full security model like the *PS* security model, but it is only secure in a weaker version of that model (the *q-bounded PS* security model) where an adversary has more limited power. Moreover, another downside of that scheme is the large *mpk* needed to describe each instance of the $(q+1,1)$-PHF; known $(q+1,1)$-PHFs need $\mathcal{O}(q^2)$ group elements [89]. Also, note that our $\mathsf{TIDNIKE}_{\mathsf{PHF}}$ construction is based on bilinear maps. In the next sections we will see how to use multilinear maps and the abstraction of the GGH candidate (see Section 4.2.2) to construct ID-NIKE schemes with the desired level of security.

4.5 Programmable Hash Functions in the Multilinear Setting

For our purposes, we will strive to construct efficient (poly, n)-PHFs for constant n (i.e. group hash functions which are $(q(k), n)$-PHFs for any polynomial q). However, there are indications that such PHFs do not exist [84], at least according to the original definition from [86]. Thus, we will adapt the definition of PHFs to the multilinear setting, and construct a "multilinear analogue" of a (poly, n)-PHF. Concretely, an (m, n)-PHF maps to a "target" group $\mathbb{G}_\ell$. Here instead of explaining $\mathsf{H}(X)$ as a product $c^{a_X} h^{b_X}$ for c, h in the target group $\mathbb{G}_\ell$ (as the case of PHFs), we will explain $\mathsf{H}(X)$ as a product $e(c_1, \ldots, c_\ell)^{a_X} e(B_X, h)$, for externally given challenges $c_i \in \mathbb{G}_1$ (which means $c = e(c_1, \ldots, c_\ell) \in \mathbb{G}_\ell$) and controlled $h \in \mathbb{G}_1$. Note that the coefficient b_X in the usual definition of a PHF now becomes a preimage $B_X \in \mathbb{G}_{\ell-1}$ under a pairing operation.

4.5.1 Definitions

Definition 4.1 (Group hash function)**.** *A group hash function* H *into* $\mathbb{G}$ *consists of two polynomial-time algorithms: the probabilistic algorithm* $\mathsf{HGen}(1^k)$ *outputs a key* hk*, and* $\mathsf{HEval}(hk, X)$ *(for a key* hk *and* $X \in \{0,1\}^k$*) deterministically outputs an image* $\mathsf{H}_{hk}(X) \in \mathbb{G}$*.*

Definition 4.2 (MPHF)**.** *Assume an* ℓ'*-group system* $\mathcal{MPG}_{\ell'} = \{\{\mathbb{G}_i\}_{i\in[\ell']}, p, \{e_{i,j}\}_{i,j\geq 1, i+j\leq \ell'}\}$ *as generated by a multilinear maps parameter generator* $\mathcal{MG}_{\ell'}(1^k)$ *(see Section 2.1.4). Let* $\mathsf{H} = (\mathsf{HGen}, \mathsf{HEval})$ *be a group hash function into* $\mathbb{G}_\ell$ *(*$\ell \leq \ell'$*), and let* $m, n \in \mathbb{N}$*. We say that* H *is an* (m, n)*-programmable hash function in the multilinear setting (*(m, n)*-MPHF) if there are polynomial-time algorithms* TGen *and* TEval *as follows.*

- $\mathsf{TGen}(1^k, c_1, \ldots, c_\ell, h)$ *(for* $c_i, h \in \mathbb{G}_1$ *and* $h \overset{\text{grp}}{\neq} 1$*) outputs a key* hk *and a trapdoor* td*. We require that for all* c_i, h*, the distribution of* hk *is statistically close to the output of* HGen*.*[5]

[5] There is a subtlety here: in case of encoded group elements, the output of TGen may consist of group elements whose noise level depends on the noise level of the c_i or h. Hence, we will assume a known a-priori bound on the noise level of the c_i and h.

- $\mathsf{TEval}(td, X)$ *(for a trapdoor* td *and* $X \in \{0,1\}^k$*) deterministically outputs* $a_X \in \mathbb{Z}$ *and* $B_X \in \mathbb{G}_{\ell-1}$ *with* $\mathsf{H}_{hk}(X) \overset{\mathsf{grp}}{=} e(c_1, \ldots, c_\ell)^{a_X} \cdot e(B_X, h)$*. We require that there is a polynomial* $p(k)$ *such that for all* hk *and* $X_1, \ldots, X_m, Z_1, \ldots, Z_n \in \{0,1\}^k$ *with* $\{X_i\}_i \cap \{Z_j\}_j = \emptyset$*,*

  $$P_{hk,\{X_i\},\{Z_j\}} := \Pr\left[a_{X_1} = \cdots = a_{X_m} = 0 \wedge a_{Z_1}, \ldots, a_{Z_n} \neq 0\right] \geq 1/p(k), \quad (4.1)$$

  where the probability is over possible trapdoors td *output by* TGen *along with the given* hk*. Furthermore, we require that* $P_{hk,\{X_i\},\{Z_j\}}$ *is close to being statistically independent of* hk*. (Formally,* $|P_{hk,\{X_i\},\{Z_j\}} - P_{hk',\{X_i\},\{Z_j\}}| \leq \nu(k)$ *for all* hk, hk' *in the range of* TGen*, all* $\{X_i\}, \{Z_j\}$*, and negligible* $\nu(k)$*.)*

We say that H *is a* $(poly, n)$*-MPHF if it is a* $(q(k), n)$*-MPHF for every polynomial* $q(k)$*, analogously for* $(m, poly)$*-MPHFs.*

Note that the TEval algorithm of an MPHF into $\mathbb{G}_1$ yields $B_X \in \mathbb{G}_0$, i.e. exponents B_X. In fact, in this case, the MPHF definition coincides with the original PHF definition from [86].

The remainder of this section is dedicated to the construction of MPHFs.

4.5.2 Programmable Random Oracles as (M)PHFs

A *programmable* random oracle H with images in $\mathbb{G}_\ell$ can be interpreted as a group hash function in the obvious way. (By "programmable", we mean that during a security proof, we can freely and adaptively determine images of H, even depending on the inputs of TGen. The only restriction of this programming is that images should appear uniformly and independently distributed to an adversary who sees only public information.) However, note for this modeling to make sense in the first place, we should require that we can hash into $\mathbb{G}_\ell$.

Theorem 4.2 (PROs as (poly, n)-(M)PHFs)**.** *A programmable random oracle* H *(in the above sense) with images in* $\mathbb{G}_\ell$ *can be programmed to act as a* $(poly, n)$*-(M)PHF for any constant* n*.*

Proof. Fix a polynomial $q = q(k)$. We show that H is a (q, n)-(M)PHF (with empty

hk). For each new preimage X, we program $\mathsf{H}(X) := e(c_1, \ldots, c_\ell)^{a_X} e(B_X, h)$ for the inputs $c_1, \ldots, c_\ell$ and h to TGen, and uniformly chosen $B_X \in \mathbb{G}_{\ell-1}$. We choose $a_X \neq 0$ with probability $1/2q$, and $a_X = 0$ otherwise. TEval outputs these a_X, B_X, assigning them as necessary for previously unqueried inputs X. For any pairwise different $X_1, \ldots, X_q, Z_1, \ldots, Z_n$, we thus have

$$\Pr\left[\forall i : a_{X_i} = 0 \wedge \forall j : a_{Z_j} \neq 0\right] = \left(1 - \frac{1}{2q}\right)^q \cdot \left(\frac{1}{2q}\right)^n \geq \frac{1}{2} \cdot \left(\frac{1}{2q}\right)^n,$$

which is significant for polynomial q and constant n. □

4.5.3 Ingredient: Efficient Admissible Hash Functions

At the heart of our construction for MPHFs lies a primitive dubbed "admissible hash function" (AHF) [31]. Unfortunately, the AHFs from Boneh and Boyen [31] are not very efficient (and in fact only achieve a weaker AHF definition, see [41]). However, an earlier work by Lysyanskaya [104] already contains an implicit and much more efficient AHF.

Intuitively, an AHF can be thought of as a combinatorial counterpart of a (poly, 1)-(M)PHF. An AHF input X is mapped to an image $\mathsf{AHF}(X)$ in such a way that in a security proof, X can fall in the set of controlled, $\mathtt{CO}$, inputs (meaning that we know a trapdoor that allows to answer adversarial queries for that input) or uncontrolled, $\mathtt{UN}$, inputs (meaning that we do not know any trapdoor but hope to embed a challenge element). Unlike with (M)PHFs, however, this is a purely combinatorial property. An AHF guarantees that for any distinct $X_1, \ldots, X_q, Z$, with significant probability, all X_i are controlled, and Z is uncontrolled.

We now give a definition that is a somewhat simpler variant of the AHF definitions from [2, 41], and then show a result implicit in [104].

Definition 4.3 (AHF)**.** *Let R be a finite set and $\ell = \ell(k)$ a polynomial in k. Let $\mathcal{AHF} = \{\mathsf{AHF} : \{0,1\}^k \to R^\ell\}$ be a family of functions. For $\mathsf{AHF} \in \mathcal{AHF}$, $K \in (R \cup \{\perp\})^\ell$, define the function $F_K : \{0,1\}^k \to \{\mathtt{CO}, \mathtt{UN}\}$ through*

$$F_K(X) = \begin{cases} \mathtt{UN} & \iff \forall i \in [\ell] : K_i = \mathsf{AHF}(X)_i \vee K_i = \perp \\ \mathtt{CO} & \iff \exists i \in [\ell] : K_i \neq \mathsf{AHF}(X)_i \wedge K_i \neq \perp, \end{cases}$$

where $\mathsf{AHF}(X)_i$ *and* K_i *denote the* i*-th component of* $\mathsf{AHF}(X)$ *and* K*, respectively.*

We say that $\mathcal{AHF}$ *is* q*-admissible if for every* $\mathsf{AHF} \in \mathcal{AHF}$*, there exists a PPT algorithm* KGen *and a polynomial* $p(k)$*, such that for all* $X_1, \ldots, X_q, Z \in \{0,1\}^k$ *with* $Z \notin \{X_i\}$*,*

$$\Pr\left[F_K(X_1) = \cdots = F_K(X_q) = \mathtt{CO} \wedge F_K(Z) = \mathtt{UN}\right] \geq 1/p(k), \tag{4.2}$$

where the probability is over $K \leftarrow \mathsf{KGen}(1^k)$*. We say that* $\mathcal{AHF}$ *is a family of admissible hash functions if* $\mathcal{AHF}$ *is* q*-admissible for all polynomials* $q = q(k)$*.*

Next, we consider a family of codes over a finite alphabet R as a family of admissible hash functions. A code over R is simply a mapping $\mathsf{C} : R^{k'} \to R^{\ell}$, where k' is the *message length* and ℓ is the *code length*. The *rate* of a code is defined as k'/ℓ. For a message X, a vector $\mathsf{C}(X)$ is called a *codeword*. The *minimum distance* of a code, $d_{\min}$, is the minimum number of positions in which any two codewords differ. This distance always satisfies $d_{\min} \leq \ell - k' + 1$, the *Singleton bound*, and codes that meet this bound are called *Maximum Distance Separable (MDS)* codes. We let a code C be represented by its parameters $(\ell, k', d_{\min})_{|R|}$. For more details on codes we refer to [23].

Theorem 4.3 ([104])**.** *Assume a family of codes* $\{\mathsf{C}_k\}$ *with* $\mathsf{C}_k : R^{k/\log_2|R|} \to R^{\ell}$ *(also interpreted as* $\mathsf{C}_k : \{0,1\}^k \to R^{\ell}$*) denoting both the code and its encoding function. Suppose that* C_k *has minimum distance at least* $c \cdot \ell$ *for a fixed constant* $c > 0$*. (That is,* $X_1 \neq X_2$ *implies that the vectors* $\mathsf{C}_k(X_1)$ *and* $\mathsf{C}_k(X_2)$ *differ in at least* $c \cdot \ell$ *positions.) Then* $\{\mathsf{C}_k\}$ *is a family of AHFs.*

Proof. Let $q = q(k)$ be a polynomial. We need to devise a PPT algorithm KGen such that (4.2) holds. $\mathsf{KGen}(1^k)$ sets $d := \left\lceil \frac{\ln(1/2q)}{\ln(1-c)} \right\rceil$ (so d is the smallest integer such that $(1-c)^d \leq 1/2q$), and picks K uniformly among all elements from $(R \cup \{\perp\})^{\ell}$ with exactly d non-$\perp$ components. Hence, the set $I := \{i \mid K_i \neq \perp\}$ is of size d.

Now fix $X_1, \ldots, X_q, Z \in \{0,1\}^k$ with $Z \notin \{X_i\}$. Our choice of K implies that $\Pr[F_K(Z) = \mathtt{UN}] = |R|^{-d}$. (Notice that K has d non-$\perp$ elements and so this is the probability that $K_i = \mathsf{AHF}(Z)_i$ for d positions.) For any fixed i, we want to upper bound the probability $\Pr[F_K(X_i) = \mathtt{UN} \mid F_K(Z) = \mathtt{UN}]$. (This step loosely

corresponds to [104, Lemma 4].) Hence, assume $F_K(Z) = \texttt{UN}$. Now $\mathsf{C}_k(X_i)$ and $\mathsf{C}_k(Z)$ differ in a set $\Delta \subseteq [\ell]$ of positions with $|\Delta| \geq c \cdot \ell$. Hence, $F_K(X_i) = \texttt{UN}$ is equivalent to $I \cap \Delta = \emptyset$. (Note that in this case $\mathsf{AHF}(Z)_j$ and $\mathsf{AHF}(X_i)_j$ do not differ for $j \in I$.) Thus,

$$\begin{aligned} \Pr\left[F_K(X_i) = \texttt{UN} \mid F_K(Z) = \texttt{UN}\right] &= \Pr\left[I \cap \Delta = \emptyset \mid F_K(Z) = \texttt{UN}\right] \\ &\leq (1-c)^d \leq \frac{1}{2q}. \end{aligned}$$

A union bound over i gives $\Pr\left[\forall i : F_K(X_i) = \texttt{UN} \mid F_K(Z) = \texttt{UN}\right] \leq 1/2$, so that

$$\Pr\left[F_K(Z) = \texttt{UN} \wedge \forall i : F_K(X_i) = \texttt{CO}\right] \geq \frac{1}{2} \cdot |R|^{-d} \geq \frac{1}{2} \cdot \left(\frac{1}{2q}\right)^{\frac{-1}{\log_{|R|}(1-c)}} = 1/p(k), \tag{4.3}$$

for $p(k) = 2 \cdot (2q)^{\frac{-1}{\log_{|R|}(1-c)}}$. Note that $1/p(k)$ grows with c. Therefore, the larger the minimum distance of the code C_k the better. □

4.5.4 Construction: MPHFs from Multilinear Maps

Our main result in this section is a simple construction of a (poly, n)-MPHF from an AHF.

Construction 4.1 (MM). *Let $\mathsf{AHF} : \{0,1\}^k \to R^\ell$ be an admissible hash function and assume an ℓ'-group system $\mathcal{MPG}_{\ell'}$. The group hash function MM from $\{0,1\}^k$ into $\mathbb{G}_\ell$ ($\ell \leq \ell'$) is given by the following algorithms:*

- $\mathsf{HGen}(1^k)$ *picks* $\tilde{h}_{i,j} \leftarrow \mathbb{G}_1 \setminus \{1\}$ *(for* $(i,j) \in [\ell] \times R$*), sets* $h_{i,j} := \mathsf{reRand}_2(\tilde{h}_{i,j})$*, and outputs* $hk := \{h_{i,j}\}_{i \in [\ell], j \in R}$.[6]
- $\mathsf{HEval}(hk, X)$ *computes* $(t_1, \ldots, t_\ell) := \mathsf{AHF}(X)$ *and outputs* $\mathsf{MM}_{hk}(X) := e(h_{1,t_1}, \ldots, h_{\ell,t_\ell})$.

Theorem 4.4. *The group hash function MM above is a $(poly, 1)$-MPHF.*

Proof. Fix a polynomial $q = q(k)$. We need to exhibit TGen and TEval algorithms as in Definition 4.2. $\mathsf{TGen}(1^k, c_1, \ldots, c_\ell, h)$ invokes $K \leftarrow \mathsf{KGen}(1^k)$ (see Definition

[6] The additional re-randomization step guarantees that the noise levels in scheme and simulation are the same. The concrete noise level of re-randomized elements depends on the maximal noise considered in the arguments of TGen.

4.3 for KGen) and, for all $(i,j) \in [\ell] \times R$ and uniform exponents $r_{i,j} \neq 0$, it sets:

$$h_{i,j} := \begin{cases} \mathsf{reRand}_2(h^{r_{i,j}}) & \text{if } K_i \neq j \text{ and } K_i \neq \perp, \\ \mathsf{reRand}_2(c_i^{r_{i,j}}) & \text{if } K_i = j \text{ or } K_i = \perp. \end{cases} \tag{4.4}$$

For now, assume $c_i \overset{\mathsf{grp}}{\neq} 1$ for all i, so our set up yields a perfectly distributed key $hk := \{h_{i,j}\}_{i,j}$ that is in fact independent of K.[7] The trapdoor is $td := ((c_i), h, K, (r_{i,j}))$.

$\mathsf{TEval}(td, X)$ computes $(t_1, \ldots, t_\ell) := \mathsf{AHF}(X)$ and distinguishes two cases:

- Case $F_K(X) = \mathtt{CO}$, i.e. there is at least an i^* with $K_{i^*} \neq t_{i^*}$ and $K_{i^*} \neq \perp$. If we set $a_X = 0$ and

  $$B_X := e(h_{1,t_1}, \ldots, h_{i^*-1,t_{i^*-1}}, h_{i^*+1,t_{i^*+1}}, \ldots, h_{\ell,t_\ell})^{r_{i^*,t_{i^*}}},$$

  for any chosen i^*, we can decompose $\mathsf{MM}_{hk}(X) \overset{\mathsf{grp}}{=} e(c_1, \ldots, c_\ell)^{a_X} e(B_X, h)$.

- Case $F_K(X) = \mathtt{UN}$, i.e. $K_i = t_i$ or $K_i = \perp$ for all i. This means that $h_{i,t_i} \overset{\mathsf{grp}}{=} c_i^{r_{i,t_i}}$ for all i, so $\mathsf{MM}_{hk}(X) \overset{\mathsf{grp}}{=} e(c_1, \ldots, c_\ell)^{a_X} e(B_X, h)$ for $a_X = \prod_i r_{i,t_i}$ and $B_X := 1$.

The AHF property (4.2) implies (4.1), for the same $p(k)$. (Note that $P_{hk,\{X_i\},\{Z\}}$ only depends on K but not on hk.)

Finally, in case $c_i \overset{\mathsf{grp}}{=} 1$ for some i, we have $e(c_1, \ldots, c_\ell) \overset{\mathsf{grp}}{=} 1$. If we replace *all* c_i in (4.4) with h, we can explain any image $\mathsf{MM}_{hk}(X) \overset{\mathsf{grp}}{=} e(h, \ldots, h)^{\prod_i r_{i,t_i}}$ as $\mathsf{MM}_{hk} \overset{\mathsf{grp}}{=} e(c_1, \ldots, c_\ell)^{a_X} e(B_X, h)$ with *arbitrary* a_X. Adjusting the probability for $a_X \neq 0$ in the order of $1/2q$ (as in the proof of Theorem 4.2) allow us to prove (4.1) for $p(k) = 4q$. □

Remark: The parameters of our MPHFs depend on the size of the alphabet R. If we use binary codes, that is $R = \{0,1\}$, with large minimum distance, we get the AHF implicit in [104]. We can obtain MPHFs that use $\ell = \mathcal{O}(k)$ groups $\mathbb{G}_i$ and have keys consisting of 2ℓ group elements. For security parameter $k = 128$, consider for example, the binary BCH code $(255, 128, 65)_2$.
Larger R give new AHFs that yield MPHFs that use fewer groups, which means

[7] In case of randomized encodings, the distribution of hk in the simulation may (e.g. with the GGH candidate) only be statistically close to the one in the scheme.

slightly less pairing-intensive constructions of MPHFs. On the other hand, larger R also implies larger keys. If we consider the AHF construction of Theorem 4.3 with MDS codes over R, we have $d_{\min} = \ell - k/\log_2|R| + 1$ which means our MPHFs use $\ell = \mathcal{O}(k/\log_2|R|)$ groups $\mathbb{G}_i$ and keys of $\ell \cdot |R| = \mathcal{O}(k \cdot |R|/\log_2|R|)$ group elements. Also, note that large $d_{\min} = c \cdot \ell$ implies $1/p(k)$ is significant (see (4.3)). Our constructions can then be instantiated with, for example, Reed-Solomon (RS) codes. For security parameter $k \approx 128$, consider for example the RS code $(31, 25, 7)_{32}$. (We note that for RS codes, in general, the size of the alphabet, $|R|$, grows with the size of the code length ℓ.) For details about about RS and BCH codes we refer to [23].

Theorem 4.5. *Let n be a constant, $q = q(k)$ be a polynomial, and let $\mathsf{H} = (\mathsf{HGen}, \mathsf{HEval})$ be a $(q + n - 1, 1)$-MPHF into $\mathbb{G}_\ell$. Then the group hash function $\mathsf{H}' = (\mathsf{HGen}', \mathsf{HEval}')$ with*

- $\mathsf{HGen}'(1^k)$ *that outputs* $hk' = (hk_\nu)_{\nu\in[n]}$ *for* $hk_\nu \leftarrow \mathsf{HGen}(1^k)$*, and*
- $\mathsf{HEval}'(hk', X)$ *that outputs* $\mathsf{H}'_{hk'}(X) := \prod_{\nu\in[n]} \mathsf{H}_{hk_\nu}(X)$

is a (q, n)-MPHF into $\mathbb{G}_\ell$.

Combining Theorems 4.4 and 4.5 yields a (poly, n)-MPHF for any constant n.

Proof of Theorem 4.5. We construct suitable TGen' and TEval' algorithms from the respective TGen and TEval algorithms for H:

- $\mathsf{TGen}'(1^k, c_1, \ldots, c_\ell, h)$ runs $(hk_\nu, td_\nu) \leftarrow \mathsf{TGen}(1^k, c_1, \ldots, c_\ell, h)$ for $\nu \in [n]$, and outputs $hk' := (hk_\nu)_{\nu\in[n]}$ and $td' := (td_\nu)_{\nu\in[n]}$.
- $\mathsf{TEval}'(hk', X)$ invokes $(a_{\nu,X}, B_{\nu,X}) \leftarrow \mathsf{TEval}(td_\nu, X)$ for $\nu \in [n]$ and outputs $a_X := \sum_{\nu\in[n]} a_{\nu,X}$ and $B_X := \prod_{\nu\in[n]} B_{\nu,X}$. This output can be justified with

$$\mathsf{H}'_{hk'}(X) \overset{\mathsf{grp}}{=} \prod_{\nu\in[n]} \mathsf{H}_{hk_\nu}(X) \overset{\mathsf{grp}}{=} \prod_{\nu\in[n]} e(c_1, \ldots, c_\ell)^{a_{\nu,X}} e(B_{\nu,X}, h) \overset{\mathsf{grp}}{=} e(c_1, \ldots, c_\ell)^{a_X} e(B_X, h).$$

Now fix $X_1, \ldots, X_q, Z_1, \ldots, Z_n$ with $\{X_i\} \cap \{Z_j\} = \emptyset$. For each ν, we hope for the following event: $a_{\nu,X_i} = 0$ for all i, and $a_{\nu,Z_j} = 0$ exactly for $j \neq \nu$. For fixed ν, due to the $(q + n - 1, 1)$ programmability of the MPHF H, this event happens with probability at least $1/p(k)$ (over td_ν) for some polynomial $p(k)$. Since $a_X = \sum_\nu a_{\nu,X}$, we get that with probability at least $(1/p(k))^n$, we have $a_{X_i} = 0$ for all i and $a_{Z_j} = a_{j,Z_j} \neq 0$ for all j (remember that for $j \neq \nu$, $a_{\nu,Z_j} = 0$), which gives us a (poly, n)-MPHF with $p'(k) = p(k)^n$. □

4.6 Fully-Secure ID-NIKE Scheme from MPHFs

In this section we revisit the ID-NIKE scheme of Sakai, Ohgishi and Kasahara (SOK) [122]. We effectively replace the random oracles with $(\text{poly}, 2)$-MPHFs in their scheme and prove the security of the generalized scheme. Using our standard-model MPHFs, this yields the first standard-model ID-NIKE scheme.[8] We then consider a hierarchical generalisation.

We assume a 2ℓ-group system $\mathcal{MPG}_{2\ell} = \{\{\mathbb{G}_i\}_{i\in[2\ell]}, p, \{e_{i,j}\}_{i,j\geq 1, i+j\leq 2\ell}\}$, generated by a multilinear maps parameter generator $\mathcal{MG}_{2\ell}(1^k)$, and a $(\text{poly}, 2)$-MPHF $\mathsf{H} = (\mathsf{HGen}, \mathsf{HEval})$ with input length in $\{0,1\}^k$ and output in $\mathbb{G}_\ell$. The component algorithms of our ID-NIKE scheme $\mathsf{IDNIKE}_{\mathsf{MPHF}}$ are defined in Figure 4.2. (Again, for compatibility with existing notation, we present an extraction algorithm Ext instead of an equivalent delegation algorithm.) Correctness of the scheme is easy to verify.[9] We now prove security.

Theorem 4.6 (Security of the MPHF-based ID-NIKE scheme)**.** *Assume* H *is a* $(poly, 2)$*-MPHF into* $\mathbb{G}_\ell$*. Then* $\mathsf{IDNIKE}_{\mathsf{MPHF}}$ *is IND-SK secure under the* $(2\ell + 1)$*-power assumption relative to generator* $\mathcal{MG}_{2\ell}$*.*

Proof. Assume an IND-SK adversary $\mathcal{A}$ against $\mathsf{IDNIKE}_{\mathsf{MPHF}}$. We construct a

[8] If we instantiate the MPHFs again with random oracles (using Theorem 4.2), we retrieve the original SOK scheme in pairing-friendly groups, along with a security proof. However, we note that our security proof uses a different, seemingly stronger computational assumption.

[9] When instantiated with the GGH candidate, the pairing e is just a multiplication. We can think of $usk_{id_1} = \mathsf{reRand}_3(\mathsf{H}_{hk}(id_1)^x)$ as $usk_{id_1} = xA + \delta_1$ and $usk_{id_2} = \mathsf{reRand}_3(\mathsf{H}_{hk}(id_2)^x)$ as $usk_{id_2} = xB + \delta_2$, for small noise values δ_1 and δ_2. We see that id_1 and id_2 are able to compute the same key: $K_{id_1,id_2} := \mathsf{ext}((xA + \delta_1)B) = xAB = \mathsf{ext}((xB + \delta_2)A) =: K_{id_2,id_1}$.

Algorithm $\mathsf{Setup}(1^k)$
$\mathcal{MPG}_{2\ell} \leftarrow \mathcal{MG}_{2\ell}(1^k)$
$x \leftarrow \mathbb{Z}_p, hk \leftarrow \mathsf{HGen}(1^k)$
$mpk := (\mathcal{MPG}_{2\ell}, hk), msk := x$
Return (mpk, msk)

Algorithm $\mathsf{Ext}(mpk, msk, id)$
$usk_{id} \leftarrow \mathsf{reRand}_3(\mathsf{H}_{hk}(id)^{msk})$
Return usk_{id}

Algorithm $\mathsf{ShK}(mpk, usk_{id_1}, id_2)$
$K_{id_1,id_2} := \mathsf{ext}(e(usk_{id_1}, \mathsf{H}_{hk}(id_2)))$
Return K_{id_1,id_2}

Figure 4.2: The ID-NIKE scheme $\mathsf{IDNIKE}_{\mathsf{MPHF}}$.

$(2\ell + 1)$-power distinguisher $\mathcal{B}$ that, given a 2ℓ-group system $\mathcal{MPG}_{2\ell}$, and group elements $g, g^x \in \mathbb{G}_1$ and $S \in \mathbb{G}_{2\ell}$, distinguishes between $S \overset{\mathsf{grp}}{=} e(g, \ldots, g)^{x^{2\ell+1}}$ (i.e. S is real), and random S.

Concretely, $\mathcal{B}$ will internally simulate $\mathcal{A}$, together with the IND-SK experiment. Let id_1^*, id_2^* be the identities from $\mathcal{A}$'s ***Test*** query (which are not known to $\mathcal{B}$ yet). Furthermore, let $q = q(k)$ be a polynomial upper bound on the total number of identities $id_i \notin \{id_1^*, id_2^*\}$ that appear in $\mathcal{A}$'s ***Extract*** and ***Reveal*** queries.[10] In the following, we will use the $(q, 2)$-MPHF property of H (and the corresponding algorithms TGen and TEval). $\mathcal{B}$ first runs $(hk, td) \leftarrow \mathsf{TGen}(1^k, \underbrace{g^x, \ldots, g^x}_{\ell \text{ times}}, g)$ and sets $mpk := (\mathcal{MPG}_{2\ell}, hk)$. Implicitly, we will have $msk := x$.

We will first describe how $\mathcal{B}$ answers an ***Extract***(id) query of $\mathcal{A}$. If $a_{id} = 0$ (for $(a_{id}, B_{id}) := \mathsf{TEval}(td, id)$), then $\mathcal{B}$ can compute $usk_{id} \leftarrow \mathsf{reRand}_3(e(B_{id}, g^x)) \overset{\mathsf{grp}}{=} \mathsf{H}_{hk}(id)^{msk}$. Otherwise, $\mathcal{B}$ aborts with output 0. We will hope for the event that $a_{id_i} = 0$ for all q identities $id_i \notin \{id_1^*, id_2^*\}$ from $\mathcal{B}$'s ***Extract*** and ***Reveal*** queries. In that case, $\mathcal{B}$ can answer not only all ***Extract*** queries from $\mathcal{A}$, but also all ***Reveal*** queries (by first computing the user private key usk_{id} of one of the two involved identities, and then using usk_{id} to compute the shared key).

[10] Note that $\mathcal{B}$ does not need to know q in advance in order to set up its simulation.

We will additionally hope for $a_{id_1^*}, a_{id_2^*} \neq 0$; in this case, $\mathcal{B}$ can embed its own challenge into the reply K^* to $\mathcal{A}$'s ***Test*** query as

$$K^* := \mathsf{ext}(S^{a_{id_1^*} a_{id_2^*}} \cdot e(B_{id_1^*}, B_{id_2^*}, g, g^x) \cdot e(B_{id_1^*}, \underbrace{g^x, \ldots, g^x}_{\ell+1 \text{ times}})^{a_{id_2^*}} \cdot e(B_{id_2^*}, \underbrace{g^x, \ldots, g^x}_{\ell+1 \text{ times}})^{a_{id_1^*}}). \quad (4.5)$$

By using $\mathsf{H}_{hk}(id_i^*) \overset{\mathsf{grp}}{=} e(\overbrace{g^x, \ldots, g^x}^{\ell \text{ times}})^{a_{id_i^*}} e(B_{id_i^*}, g)$, we see that $K^* = \mathsf{ext}(e(\mathsf{H}_{hk}(id_1^*)^x, \mathsf{H}_{hk}(id_2^*))) = K_{id_1^*, id_2^*}$ whenever $S \overset{\mathsf{grp}}{=} e(\underbrace{g^x, \ldots, g^x}_{2\ell \text{ times}})^x$. Conversely, if S is random, then so is K^*. (If $a_{id_i^*} = 0$ for some $i \in \{1, 2\}$, then $\mathcal{B}$ aborts with output 0.)

Finally, $\mathcal{B}$ outputs $\hat{b}$ (i.e. $\mathcal{A}$'s guess for the bit b in the IND-SK experiment). If the event that $\mathcal{B}$ aborts is independent of the queried identities id_i and id_i^* (as is the case for the RO-based MPHF from Theorem 4.2), we have

$$|\Pr[\mathcal{B} = 1 \mid S \text{ real}] - \Pr[\mathcal{B} = 1 \mid S \text{ random}]| = \Pr[\neg\mathsf{abort}] \cdot \mathrm{Adv}^{\text{IND-SK}}_{\mathcal{A}, \mathsf{IDNIKE}_{\mathsf{MPHF}}}(k).$$

Hence, $\mathcal{B}$ breaks the $(2\ell+1)$-power assumption if and only if $\mathcal{A}$ breaks the IND-SK security of $\mathsf{IDNIKE}_{\mathsf{MPHF}}$.

However, in the general case, abort might not be independent of the id_i and id_i^*. This is because $\Pr[\mathsf{abort}]$ is upper bounded by $1 - 1/p(k)$ (for the polynomial $p(k)$ from (4.2)) and this means that different sets of identities id_i, id_i^* may cause the simulator to abort with different probabilities (smaller than $1 - 1/p(k)$ of course). Hence, in order to make $\Pr[\mathsf{abort}]$ independent of the adversary's queries, we will have to resort to an "artificial abort" strategy as in [138]. The artificial abort step consists of, at the end of a successful simulation a) estimating the probability that the queries made by $\mathcal{A}$ cause $\mathcal{B}$ to abort, and then b) if necessary "artificially aborting" so that $\mathcal{B}$ aborts with maximum probability, $1 - 1/p(k)$. That is, even if $a_{id_i} = 0$ and $a_{id_i^*} \neq 0$ for all i, $\mathcal{B}$ will "artificially" abort with probability $1 - 1/(P_{(id_i),(id_i^*)} \cdot p(k))$ for $P_{(id_i),(id_i^*)} := \Pr[\neg\mathsf{abort} \mid (id_i), (id_i^*)]$. This keeps the (new) abort probability at $1 - 1/p(k)$, independently of the id_i and id_i^*, and enables an analysis as above. Unfortunately, in the general case, we can only approximate $P_{(id_i),(id_i^*)}$ (up to an inverse polynomial error, by running TEval with freshly generated keys sufficiently often), which introduces an additional error term in the analysis. We refer to [138] for details on the artificial abort technique. □

A variant secure under a weaker assumption. We can also construct an ID-NIKE scheme in the standard model using two instances (with keys hk_1, hk_2) of a $(\text{poly}, 1)$-MPHF instead of a single instance of a $(\text{poly}, 2)$-MPHF. Shared keys are computed as $K := \mathsf{ext}(e(\mathsf{H}_{hk_1}(id_1)^{msk}, \mathsf{H}_{hk_2}(id_2)))$; user private keys are of the form $usk_{id} = (\mathsf{reRand}_3(\mathsf{H}_{hk_1}(id)^{msk}), \mathsf{reRand}_3(\mathsf{H}_{hk_2}(id)^{msk}))$. The benefit of this variant is that it is possible to prove security under the 2ℓ-MDDH assumption (as opposed to the potentially stronger $(2\ell+1)$-power assumption we use above). The proof is similar to the one above; however, we will hope that $a_{1,id_i} = a_{2,id_i} = 0$ for all non-challenge queries id_i, and that $a_{1,id_1^*}, a_{2,id_2^*} \neq 0$ and $a_{1,id_2^*} = a_{2,id_1^*} = 0$, where $(a_{j,id}, B_{j,id}) = \mathsf{TEval}(td_j, id)$ $(j \in \{1,2\})$.

4.6.1 Extension to Hierarchical ID-NIKE (H-ID-NIKE)

We can extend our ID-NIKE scheme to an H-ID-NIKE scheme of constant depth L. To this end, we work in a $2\ell L$-group system $\mathcal{MPG}_{2\ell L}$, and use L instances of a $(\text{poly}, 2)$-MPHF H into $\mathbb{G}_\ell$.

The resulting H-ID-NIKE scheme, denoted by $\mathsf{HIDNIKE}_{\mathsf{MPHF}}$, is given in Figure 4.3. In that description, and in the following, we write $\mathbf{id}_{\lceil i} := (id_1, \ldots, id_i)$ for an identity $\mathbf{id} = (id_1, \ldots, id_d)$ and $i \leq d$. We assume that all involved identities (including "shortened identities" $\mathbf{id}_{\lceil i}$) can be uniquely encoded as k-bit strings. (If this is not the case, we can always first apply a collision-resistant hash function.) In general, for identities $\mathbf{id}, \mathbf{id}_j$ we let d, d_j denote their depth in the hierarchy, respectively.

Theorem 4.7 (Security of the MPHF-based H-ID-NIKE scheme)**.** *Let* H *be a* $(poly, 2)$*-MPHF into* $\mathbb{G}_\ell$*. For fixed depth* $L \in \mathbb{N}$, $\mathsf{HIDNIKE}_{\mathsf{MPHF}}$ *is secure under the* $(2\ell L + 1)$*-power assumption relative to generator* $\mathcal{MG}_{2\ell L}$*.*

Proof. The proof is very similar to the proof of Theorem 4.6; we focus on the necessary adaptations. We construct a $(2\ell L + 1)$-power distinguisher $\mathcal{B}$ from an IND-SK adversary $\mathcal{A}$. Assume $\mathcal{B}$ gets a $2\ell L$-group system $\mathcal{MPG}_{2\ell L}$ and group elements $g, g^x \in \mathbb{G}_1$ and $S \in \mathbb{G}_{2\ell L}$ as input, and is supposed to distinguish the cases $S \overset{\mathsf{grp}}{=} e(g^x, \ldots, g^x)^x$ and random S.

$\mathcal{B}$ simulates the IND-SK experiment for $\mathcal{A}$. First, $\mathcal{B}$ runs $(hk_i, td_i) \leftarrow \mathsf{TGen}(1^k,$

Algorithm $\mathsf{Setup}(1^k, L)$

$\mathcal{MPG}_{2\ell L} \leftarrow \mathcal{MG}_{2\ell L}(1^k)$

$x \leftarrow \mathbb{Z}_p$, $\tilde{u} \leftarrow \mathbb{G}_\ell, u \leftarrow \mathsf{reRand}_2(\tilde{u})$, $hk_i \leftarrow \mathsf{HGen}(1^k)$ $(i \in [L])$

$mpk := (\mathcal{MPG}_{2\ell L}, \{hk_i\}_{i\in[L]}, u)$, $msk := x$

Return (mpk, msk)

Algorithm $\mathsf{Del}(mpk, usk_{\mathbf{id}}, \mathbf{id}')$

Parse $\mathbf{id}' =: (id_1, \ldots, id_{d+1})$

If $\mathbf{id} \neq (id_1, \ldots, id_d)$ return $\perp$

$usk_{\mathbf{id}'} \leftarrow \mathsf{reRand}_{d+3}(e(usk_{\mathbf{id}}, \mathsf{H}_{hk_{d+1}}(\mathbf{id}')))$

Return $usk_{\mathbf{id}'}$

Algorithm $\mathsf{ShK}(mpk, usk_{\mathbf{id}_1}, \mathbf{id}_2)$

$Y_{\mathbf{id}_2} := e(\mathsf{H}_{hk_1}(\mathbf{id}_{2,\lceil 1}), \ldots, \mathsf{H}_{hk_{d_2}}(\mathbf{id}_{2,\lceil d_2}))$, $\mathbf{u} := \underbrace{u, \ldots, u}_{2L-d_1-d_2 \text{ times}}$

$K_{\mathbf{id}_1,\mathbf{id}_2} := \mathsf{ext}(e(usk_{\mathbf{id}_1}, Y_{\mathbf{id}_2}, \mathbf{u}))$

Return $K_{\mathbf{id}_1,\mathbf{id}_2}$

Figure 4.3: The H-ID-NIKE scheme $\mathsf{HIDNIKE}_{\mathsf{MPHF}}$.

Note. $msk = usk_\varepsilon = x \in \mathbb{Z}_p = \mathbb{G}_0$, so Del can be used to derive level-1 user private keys from msk. (Recall that our definition of e is consistent with the implicit exponent group $\mathbb{G}_0 = \mathbb{Z}_p$; e.g. $e(x, g) = g^x$ for $x \in \mathbb{G}_0$.)

$\underbrace{g^x, \ldots, g^x}_{\ell \text{ times}}, g)$ $(i \in [L])$ and then for $u \leftarrow \mathsf{reRand}_2(e(\underbrace{g^x, \ldots, g^x}_{\ell \text{ times}}))$, it sets $mpk := (\mathcal{MPG}_{2\ell L}, \{hk_i\}_{i\in[L]}, u)$. Implicitly we will have $msk := x$. To answer an ***Extract*** query for identity $\mathbf{id} = (id_1, \ldots, id_d)$, $\mathcal{B}$ will hope for $a_{\mathbf{id}} := \prod_{i\in[d]} a_{i,\mathbf{id}_{\lceil i}} = 0$, where $(a_{i,\mathbf{id}_{\lceil i}}, B_{i,\mathbf{id}_{\lceil i}}) := \mathsf{TEval}(td_i, \mathbf{id}_{\lceil i})$. In that case, we must have $a_{i^*,\mathbf{id}_{\lceil i^*}} = 0$ for some i^*, and thus $\mathcal{B}$ can compute $usk_{\mathbf{id}}$ using

$$usk_{\mathbf{id}} \leftarrow \mathsf{reRand}_{d+2}(e(\mathsf{H}_{hk_1}(\mathbf{id}_{\lceil 1}), \ldots, \mathsf{H}_{hk_{i^*-1}}(\mathbf{id}_{\lceil i^*-1}), e(B_{i^*,\mathbf{id}_{\lceil i^*}}, g^x), \mathsf{H}_{hk_{i^*+1}}(\mathbf{id}_{\lceil i^*+1}), \ldots, \mathsf{H}_{hk_d}(\mathbf{id}_{\lceil d}))) \overset{\mathsf{grp}}{=} e(\mathsf{H}_{hk_1}(\mathbf{id}_{\lceil 1}), \ldots, \mathsf{H}_{hk_d}(\mathbf{id}_{\lceil d}))^{msk}.$$

Conversely, $\mathcal{B}$ can embed its own challenge S into the challenge key K^* whenever the challenge identities $\mathbf{id}_1^* = (id_{1,1}^*, \ldots, id_{1,d_1^*}^*)$ and $\mathbf{id}_2^* = (id_{2,1}^*, \ldots, id_{2,d_2^*}^*)$ satisfy $a_{\mathbf{id}_1^*}, a_{\mathbf{id}_2^*} \neq 0$. Namely, in that case, the group element

$$e(\mathsf{H}_{hk_1}(\mathbf{id}_{1,\lceil 1}^*), \ldots, \mathsf{H}_{hk_{d_1^*}}(\mathbf{id}_{1,\lceil d_1^*}^*), \mathsf{H}_{hk_1}(\mathbf{id}_{2,\lceil 1}^*), \ldots, \mathsf{H}_{hk_{d_2^*}}(\mathbf{id}_{2,\lceil d_2^*}^*), \underbrace{u, \ldots, u}_{2L-d_1^*-d_2^* \text{ times}})^{msk}, \tag{4.6}$$

from which the shared key is computed, contains a factor of the form $e(\overbrace{g^x, \ldots, g^x}^{2\ell L \text{ times}})^x$, which can be replaced by B's own challenge S; the remaining $2^{d_1^*+d_2^*} - 1$ factors of (4.6) can be computed as in (4.5).

Hence, $\mathcal{B}$'s simulation requires that $a_{\mathbf{id}} = 0$ for all non-challenge queried identities $\mathbf{id}$, and $a_{\mathbf{id}_1^*}, a_{\mathbf{id}_2^*} \neq 0$ for challenge identities $\mathbf{id}_1^*, \mathbf{id}_2^*$. It will be sufficient to hope for $a_{\mathbf{id}_{j,\lceil i}^*} \neq 0$ ($j \in \{1,2\}$) for all prefixes of the challenge identities, and $a_{\mathbf{id}_{\lceil i}} = 0$ for all other involved prefixes. (Since no prefixes of the challenge identities will need to be extracted, these requirements are not contradictory.) These requirements translate to requirements on the L individual MPHF instances. Hence, with probability at least $(1/p(k))^L$ (for the polynomial $p(k)$ from (4.1)), the simulation will not abort.

The remaining analysis (including a necessary artificial abort step) can be performed as in the proof of Theorem 4.6. □

A more efficient variant in the random oracle model. We can replace the $2\ell L$-group system with a $2L$-group system and the L different MPHFs with a random oracle hashing into $\mathbb{G}_1$ in the above scheme $\mathsf{HIDNIKE}_{\mathsf{MPHF}}$ to obtain a second H-ID-NIKE scheme which can be proven secure in the random oracle model. In this case, the $2L$-group system can be instantiated with smaller parameters than the $2\ell L$-group system required in our standard model scheme.

4.7 Final Notes

In this chapter, we focused on the construction of ID-NIKE schemes. We started our contributions in the area by giving a construction towards the first ID-NIKE scheme with security proof in the standard model. Our construction, $\mathsf{TIDNIKE}_{\mathsf{PHF}}$, is very simple: it only uses a pairing and two instantiations of a $(q+1,1)$-PHF. However, the security of $\mathsf{TIDNIKE}_{\mathsf{PHF}}$ relies on the assumption that an adversary uses at most q non-challenge identities in its oracle queries. (Achieving full security using a $(q+1,1)$-PHF was not possible due to the inexistence of $(\text{poly},1)$-PHFs [84].)

Then, we adapted the definition of a PHF, working in a setting where multilinear maps are available, and constructed analogues of (poly, n)-PHFs, which we called (poly, n)-MPHFs. Using our (poly, n)-MPHFs we were then able to construct the first fully secure ID-NIKE scheme with security in the standard model, $\mathsf{IDNIKE}_{\mathsf{MPHF}}$. Additionally, we were able to construct the first fully secure hierarchical ID-NIKE with security either in the standard model or in the random oracle model, $\mathsf{HIDNIKE}_{\mathsf{MPHF}}$.

As future work, it would be very interesting to formally extend our constructions to the more general setting of multiple and independent trusted authorities as well as to the setting where shared keys can be computed by groups of users instead of just pairs of users. In the next subsection we expand more on these ideas.

4.7.1 (H-)ID-NIKE with Multiple TAs and Group (H-)ID-NIKE

Consider a scenario where we have multiple, independent TAs (sharing some common system parameters) each issuing private keys to users, and we wish to enable any pair of users with private keys issued by possibly different TAs to be able to compute a shared key. Our Setup algorithm from Section 4.3 would now be divided into two algorithms: one that generates a set of common parameters *params*, and is run by a large organization which then makes these parameters public; and another, run independently by any TA, that on input *params* outputs a master public key *mpk* and a master secret key *msk*. Note that ID-NIKE with multiple TAs is different from H-ID-NIKE. In the latter a single root TA generates the public parameters and a master secret key from which the private keys (master secret keys) of the sub-TAs are obtained. In the scenario of ID-NIKE with multiple independent TAs, there is no root TA and each TA independently generates its pair of master public key and master secret key. This scenario would also be possible in the hierarchical setting, H-ID-NIKE, which would involve multiple independent root TAs.

Going further, we may wish to enable groups of users (rather than just pairs of users) in the "forest" of hierarchies to compute shared keys. All of this is enabled in the multilinear setting by generalisation of our ID-NIKE and H-ID-NIKE schemes. For simplicity, we sketch just one such scheme here, leaving detailed development of

these ideas to future work.

Suppose we have a 3-group system and let H be a random oracle with outputs in $\mathbb{G}_1$. Then we can instantiate our ID-NIKE scheme with the MPHF being replaced by H and with private keys of the form $usk_{id,i} \leftarrow \mathsf{reRand}_3(\mathsf{H}(id)^{msk_i}) \in \mathbb{G}_1$. Now assume we have two trusted authorities TA_1, TA_2 with master secrets msk_1, msk_2. We augment mpk_i to include the element $h_i = \mathsf{reRand}_2(g^{msk_i})$. Consider the chain of equalities:

$$\begin{aligned} e(usk_{id_1,1}, \mathsf{H}(id_2), h_2) &\stackrel{\mathsf{grp}}{=} e(\mathsf{H}(id_1)^{msk_1}, \mathsf{H}(id_2), g^{msk_2}) \\ &\stackrel{\mathsf{grp}}{=} \ldots \stackrel{\mathsf{grp}}{=} e(usk_{id_2,2}, \mathsf{H}(id_1), h_1). \end{aligned}$$

The first computation in the chain can be carried out by user id_1 using its private key issued by TA_1, while the last can be done by id_2 using its private key issued by TA_2; thus the output can be used as the basis of a shared key (by applying ext in the usual way). Hence two users with private keys issued by different and possibly *independent* TAs can still compute a shared key non-interactively. We leave the generalization of this simple scheme a) to the standard model, b) to greater numbers of users and TAs, and c) to the hierarchical setting, to future work.

Part II

Hierarchical Key Assignment Schemes

Chapter 5

Hierarchical Key Assignment Schemes (HKAS)

Contents

Hierarchical key assignment schemes can be used to enforce access control policies by cryptographic means. In this chapter, we present new enhanced security models for such schemes. We also give simple, efficient, and strongly-secure constructions for hierarchical key assignment schemes for arbitrary hierarchies using PRFs and FS-PRGs. We compare instantiations of our constructions with state-of-the-art hierarchical key assignment schemes, demonstrating that our new schemes possess an attractive trade-off between storage requirements and efficiency of key derivation. Most of the content of this chapter appears in [65], which is joint work with Kenneth G. Paterson and Bertram Poettering. Some of the results generalize part of [62], which is joint work with Kenneth G. Paterson.

5.1 Introduction

A hierarchical key assignment scheme (HKAS) is a method for implementing *access control policies* by assigning encryption keys and private information to each class in a hierarchy in such a way that the private information assigned to a class, along with some public information, can be used to derive encryption keys to all classes lower down in the hierarchy. Formally, the hierarchy is modelled as a *partially ordered set (poset)*, each data item is labelled by a class u in the hierarchy, and is encrypted using the encryption key κ_u corresponding to that class. Now a user, given access to the private information S_u, can derive the relevant encryption key κ_v for any descendant class v, and hence gain access to the data of class v.

Such hierarchical key assignment schemes can be used in many applications where it is desirable to provide differentiated access to data according to an access control policy. As an illustration, consider the management of a database containing sensitive information, such as medical records in a hospital; doctors, depending on their seniority, are assigned access permission to a set of files in a patient's medical record, while nurses, being at a lower level in the hierarchy, have more restricted access to that information. Other application domains include government communication, protection of industrial secrets, and broadcast services such as cable TV.

The use of cryptographic techniques to solve the problem of key management in hierarchical structures was first proposed in 1983, by Akl and Taylor [5], who put forward the concept of a hierarchical key assignment scheme. Since then, a large number of different schemes have been proposed, offering different trade-offs in terms of the amount of public and private storage required and the complexity of key derivation – see for example [8–11, 45, 48, 50, 85, 105, 123, 127, 134, 137, 141, 142]. Many additional issues are addressed in these works: time-dependent constraints, dynamic addition and removal of classes, and revocation, for example. A recent survey of this area by Crampton *et al.* [47] provides a detailed classification and analyses of many of the schemes proposed in the last decades.

Many of the early hierarchical key assignment schemes lacked any formal security analysis, but this shortcoming has been gradually addressed beginning with the work of Atallah *et al.* [8], who proposed two different security notions: security against key

recovery attacks (KR-security) and security with respect to key indistinguishability (KI-security). Informally, KR-security captures the notion that an adversary should not be able to compute a key to which it should not have access; whereas in the notion of KI-security, the adversary should not even be able to distinguish between the real key and a random string of the same length. The stronger KI-security notion is important in enabling secure composability for hierarchical key assignment schemes, that is, in achieving the property that any secure key assignment scheme can be safely used alongside any suitably secure encryption scheme.

5.1.1 Our Contributions

We argue that the security notions for hierarchical key assignment schemes introduced in [8] need to be strengthened in order to capture the widest possible range of realistic attacks. In particular, the KR and KI security notions by Atallah *et al.* do not allow an adversary to gain access to encryption keys κ_v for classes above the target class u, even though these encryption keys might leak through usage and their compromise need not directly lead to a compromise of the private information S_u or encryption key κ_u for the target class. Thus, as an initial contribution in the area, we first define strengthened security models that provide this additional compromise capability to the adversary, and show that our new models are strictly stronger than the corresponding KR and KI security notions. Section 5.2.1 contains the details.

We next propose two very simple and efficient hierarchical key assignment schemes for arbitrary posets, and prove them to be secure in the sense of our strengthened KI-security notion. Both of our schemes exploit the *chain partition* idea recently introduced by Crampton *et al.* [48]. This gives a method of constructing an HKAS for an arbitrary access structure, modelled as a poset P, from an HKAS for a simple chain C (i.e. an HKAS for a totally ordered set). This is done by partitioning the poset into chains and building the keys for the more complex scheme in a particular way from the keys of the simpler chains. This interesting approach was proposed without any formal security analysis in [48]. We provide in Section 5.3 a generic security analysis of the chain partition construction, showing that the security of the resulting scheme for a poset P in our strengthened KI security model is equivalent

to the security (also in our strengthened KI security model) of the scheme for a single chain. (It will be evident that our security analysis can be slightly modified to also show the equivalence of security of the two schemes in other security models.) It is worth noting that this construction can support different levels of security or efficiency of key derivation for different subgroups in a hierarchy, by using different schemes in each chain.

The chain partition construction enables us to focus on constructing efficient HKAS for totally ordered sets in our strengthened KI security model. Our first construction, presented in Section 5.4.1, is based only on pseudorandom functions (PRFs), which can be efficiently implemented using, for example, HMAC [20] built using only a cryptographic hash function. Our second construction, presented in Section 5.4.2, is based on any forward-secure pseudorandom generator (FS-PRG). The instantiation of our latter construction with the BBS FS-PRG from Construction 2.1 yields an HKAS with security under the factoring assumption. We note that an FS-PRG can also be obtained cheaply and generically from any PRG using the constructions of Bellare and Yee [19].

In Section 5.5 we provide a comparison of instantations of our new constructions with a variety of proven-secure HKASs from the literature.

5.2 Hierarchical Key Assignment Schemes

A partially ordered set (poset) is a pair $(V, \preceq)$ where V is a finite set of pairwise disjoint classes, called *security classes*, and '$\preceq$' is a partial order on V, i.e. is a reflexive, antisymmetric, and transitive binary relation. A security class can represent a person, a department, or a user group in an organisation. The relation $\preceq$ is defined in accordance with authority for each class in V: for any two classes $u, v \in V$ we write $v \preceq u$ or $u \succeq v$ to indicate that users in class u can access the data of users in class v. We say that u *covers* v, denoted $v \lessdot u$ or $u \gtrdot v$, if $v \prec u$ and there does not exist $c \in V$ such that $v \prec c \prec u$. $(V, \preceq)$ is a *totally ordered set* (or *chain*) if for all $u, v \in V$, either $v \prec u$ or $u \prec v$ or $u = v$. We say that $A \subseteq V$ is an *antichain* in V if for all $u, v \in A, u \neq v$, we have $v \not\preceq u$ and $v \not\succeq u$. Any poset $(V, \preceq)$ can be represented by a specific directed acyclic graph $G = (V, E)$, called the *access graph*, where

the vertices coincide with the security classes and there is an edge from class u to class v if and only if $u \succ v$. A *partition* of a set V is a collection of sets $\{V_1, \ldots, V_s\}$ such that (i) $V_i \subseteq V \; \forall i$, (ii) $V_1 \cup \ldots \cup V_s = V$, and (iii) $i \neq j \Rightarrow V_i \cap V_j = \emptyset$.

The problem of key management for such posets consists of assigning private information and encryption keys (e.g. to be used in a symmetric encryption scheme) to each class in the poset in such a way that the encryption keys can be used to protect or access data, whereas the private information can be used to efficiently derive the keys for any descendant class in the poset. The cryptographic primitive that solves this challenge is called a hierarchical key assignment scheme [5], and is defined as follows.

Definition 5.1 (Hierarchical Key Assignment Scheme)**.** *Let* Γ *denote a set of access graphs, i.e. of graphs that correspond to posets. A* hierarchical key assignment scheme (HKAS) *for* Γ *is a pair of algorithms* $(\mathsf{Gen}, \mathsf{Der})$ *satisfying the following conditions:*

- $\mathsf{Gen}(1^k, G)$ *is a probabilistic polynomial-time algorithm that takes as input a security parameter* 1^k *and a graph* $G = (V, E) \in \Gamma$ *and outputs*
  - *for all classes* $u \in V$*: private information* S_u *and key* $\kappa_u \in \{0,1\}^{p(k)}$*, for a fixed polynomial* p*;*
  - *public information pub.*

  We denote by (S, κ, pub) *the output of* $\mathsf{Gen}(1^k, G)$*, where* $S = (S_u)_{u \in V}$ *and* $\kappa = (\kappa_u)_{u \in V}$ *are the vectors of private information and keys, respectively.*
- $\mathsf{Der}(G, u, v, S_u, pub)$ *is a deterministic polynomial-time algorithm that takes as input a graph* G*, classes* $u, v \in V$ *such that* $v \preceq u$*, private information* S_u*, and public information pub, and outputs a key* $\kappa_v \in \{0,1\}^{p(k)}$ *assigned to class* v*.*

For correctness we require that for all $k \in \mathbb{N}$, all $G \in \Gamma$, all (S, κ, pub) output by $\mathsf{Gen}(1^k, G)$, and all $u, v \in V, v \preceq u$, we have $\mathsf{Der}(G, u, v, S_u, pub) = \kappa_v$.

Remark: Observe that hierarchical key assignment schemes are essentially symmetric in nature, i.e. a separation of entities holding secret keys and entities holding

public keys is not assumed. As a consequence, Gen's output *pub* does not have to be public in the classical sense, but could also be folded into the private information S_u, for all $u \in V$. We point out that we left *pub* in Definition 5.1 to keep it consistent with prior work. However, the schemes that we propose in this thesis will not make use of *pub*, i.e. they will assign an empty value to it.

5.2.1 Definitions of Security for HKAS

As mentioned earlier, formal security modelling for hierarchical key assignment schemes began with [8]. However, as we will argue, these models are inadequate for practical application in the most challenging of security environments. In the following, we describe our new, strengthened models, and then discuss the differences with respect to the established ones.

We consider strengthened versions of the key indistinguishability (KI) and key recovery (KR) security goals proposed by Atallah *et al.* [8], which we name strong key indistinguishability (S-KI) and strong key recovery (S-KR) respectively. For each security notion we consider both static and dynamic adversaries. It will shortly become clear, however, that security against static adversaries is polynomially equivalent to security against dynamic adversaries. We begin with informal statements of our security models, and then give formal models in terms of security experiments involving an adversary.

Strong key indistinguishability. We consider two types of adversaries:

- Static adversaries: A static adversary $\mathcal{A}_{stat}$, given an access graph $G = (V, E)$, first chooses a security class $u \in V$ to attack. Using algorithm Gen on graph G, the experiment generates (S, κ, pub). The adversary is then provided with private information S_v assigned to all classes $v \in V$ such that $v \not\succeq u$ (i.e. the adversary is given all private information S_v that should *not* enable the computation of key κ_u) along with the set of all keys κ_v associated with classes $v \in V$ such that $v \succ u$, and the public information *pub*. Precisely, the adversary gets *pub* and the two sets:

$$Corrupt_{G,S,u} = \{S_v \in S \mid v \not\succeq u\} \qquad \text{and} \qquad Keys_{G,u} = \{\kappa_v \mid v \succ u\}\,.$$

Notice that, given $Corrupt_{G,S,u}$, the adversary can compute for itself all keys κ_v for classes $v \in V$ such that $v \not\geq u$. So, from the obtained information, the adversary can gain access to κ_v for any $v \in V \setminus \{u\}$. As a challenge, the adversary additionally gets either key κ_u or a random string of the same length, and it has to distinguish these two cases. We refer to Definition 5.2 for the formal specification of this experiment.

- Dynamic adversaries: In contrast to a static adversary, a dynamic (also called adaptive) adversary $\mathcal{A}_{dyn}$ first gets access to all public information pub and then may request keys κ_v and private information S_v in an adaptive manner before eventually committing to a security class $u \in V$ that it wants to attack. After receiving a challenge based on key κ_u, it continues to request keys and private information until terminating and outputting a bit. The adversary wins the security experiment if it successfully distinguishes the key κ_u from random, under the restriction that (i) $u \not\leq v$ for all classes v for which the adversary requested private information, and (ii) key κ_u has not been requested.

Strong key recovery. Again, we consider two types of adversaries:

- Static adversaries: The difference between security in the sense of strong key recovery with respect to static adversaries (S-KR-ST) and strong key indistinguishability with respect to static adversaries (S-KI-ST) is that in the former the adversary receives no challenge key, and the adversary is required to recover the key κ_u corresponding to the attacked class u rather than to distinguish it from a random key. We refer to Definition 5.3 for the formal specification of this experiment.
- Dynamic adversaries: Same as above; here the adversary never receives a challenge key. Instead, the adversary is required to recover the key κ_u corresponding to the chosen class to attack, u.

It is not difficult to see that, both in the S-KI notion and in the S-KR notion, security against static adversaries is actually polynomially equivalent to security against dynamic adversaries. Indeed, in the corresponding reduction, the static adversary simply guesses which class will be the subject of the dynamic adversary's query, and aborts if the guess turns out to be incorrect; this reduction succeeds with probability $1/|V|$. A similar proof was used in [10] (and implicitly in [8]).

So schemes proven secure against static adversaries are automatically also secure against dynamic adversaries (albeit with a less tight overall security reduction). In the remainder of this chapter, we focus on the static case.

We next give our definitions for security in the sense of strong key indistinguishability with respect to static adversaries (S-KI-ST-security) and strong key recovery with respect to static adversaries (S-KR-ST-security), formalising the above discussion.

Definition 5.2 (S-KI-ST)**.** *Let* Γ *be a set of access graphs and let* $(\mathsf{Gen}, \mathsf{Der})$ *be a hierarchical key assignment scheme for* Γ*. Consider the following experiment associated with adversary* $\mathcal{A}$*:*

$$\begin{array}{l}\text{Experiment } \mathrm{Exp}^{\text{S-KI-ST},\,d}_{\mathcal{A},\Gamma}(1^k):\\ \quad u \leftarrow \mathcal{A}(1^k, G)\\ \quad (S, \kappa, pub) \leftarrow \mathsf{Gen}(1^k, G)\\ \quad \textit{If } d = 1 \textit{ then } T \leftarrow \kappa_u \textit{ else } T \leftarrow \{0,1\}^{p(k)}\\ \quad d' \leftarrow \mathcal{A}(pub, \mathit{Corrupt}_{G,S,u}, \mathit{Keys}_{G,u}, T)\\ \quad \textit{Return } d'\end{array}$$

For any $G \in \Gamma$*, the advantage of* $\mathcal{A}$ *in the above experiment is defined as*

$$\mathrm{Adv}^{\text{S-KI-ST}}_{\mathcal{A},\Gamma}(k) = 2\left|\Pr\left[\mathrm{Exp}^{\text{S-KI-ST},\,d}_{\mathcal{A},\Gamma}(1^k) = d : d \leftarrow \{0,1\}\right] - 1/2\right|,$$

which can also be written as

$$\mathrm{Adv}^{\text{S-KI-ST}}_{\mathcal{A},\Gamma}(k) = \left|\Pr[\mathrm{Exp}^{\text{S-KI-ST},1}_{\mathcal{A},\Gamma}(1^k) = 1] - \Pr[\mathrm{Exp}^{\text{S-KI-ST},0}_{\mathcal{A},\Gamma}(1^k) = 1]\right|.$$

The key assignment scheme is said to be secure in the sense of strong key indistinguishability with respect to static adversaries (S-KI-ST-secure) *if* $\mathrm{Adv}^{\text{S-KI-ST}}_{\mathcal{A},\Gamma}(k)$ *is negligible for every polynomial-time adversary* $\mathcal{A}$ *and any graph* $G \in \Gamma$*.*

Definition 5.3 (S-KR-ST)**.** *Let* Γ *be a set of access graphs and let* $(\mathsf{Gen}, \mathsf{Der})$ *be a hierarchical key assignment scheme for* Γ*. Consider the following experiment associated with adversary* $\mathcal{A}$*:*

$$\begin{array}{l}\text{Experiment } \mathrm{Exp}^{\text{S-KR-ST}}_{\mathcal{A},\Gamma}(1^k):\\ \quad u \leftarrow \mathcal{A}(1^k, G)\\ \quad (S, \kappa, pub) \leftarrow \mathsf{Gen}(1^k, G)\\ \quad \kappa'_u \leftarrow \mathcal{A}(pub, \mathit{Corrupt}_{G,S,u}, \mathit{Keys}_{G,u})\\ \quad \textit{If } \kappa'_u = \kappa_u \textit{ return } 1 \textit{ else return } 0\end{array}$$

For any $G \in \Gamma$*, the advantage of* $\mathcal{A}$ *in the above experiment is defined as*

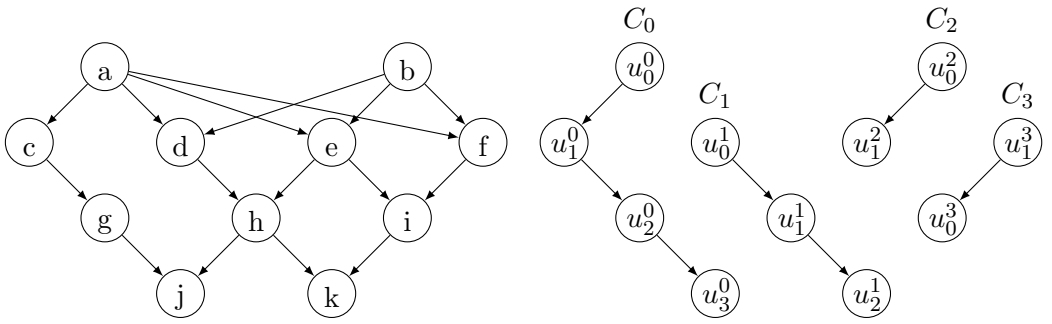
$$\mathrm{Adv}^{\text{S-KR-ST}}_{\mathcal{A},\Gamma}(k) = \left|\Pr\left[\mathrm{Exp}^{\text{S-KR-ST}}_{\mathcal{A},\Gamma}(1^k) = 1\right]\right|.$$

The key assignment scheme is said to be secure in the sense of strong key recovery with respect to static adversaries (S-KR-ST-secure) *if* $\mathrm{Adv}^{\text{S-KR-ST}}_{\mathcal{A},\Gamma}(k)$ *is negligible for every polynomial-time adversary* $\mathcal{A}$ *and any graph* $G \in \Gamma$.

We now explain why our security models are stronger than the ones introduced by Atallah *et al.* [8]. While an adversary in our S-KI-ST or S-KR-ST security model receives both the set $Corrupt_{G,S,u} \subseteq S$ of private information and the set $Keys_{G,u} \subseteq \{0,1\}^{p(k)}$ of (symmetric) keys, in the KI-ST and KR-ST security models from [8] the adversary receives only the former set when performing its attack. In our dynamic settings, the strong adversary has access to keys κ_v for which $v \succ u$, where u is the challenge security class, whereas in the dynamic models of [8], the adversary has no access to such keys. Now in a real deployment of a scheme, some of the cryptographic keys κ_v used in the scheme may leak, perhaps through cryptanalysis or misuse. In this case, we would like our selected security model to provide the strongest possible guarantees about the security of other keys that have not been leaked. But note that the previous security models from [8] provide no such guarantees, whereas our models provide the strongest possible guarantee, in that *all* keys κ_v with $v \succ u$ are given to the adversary. Indeed, as the next example makes clear, it is quite feasible that leakage of a key κ_v for which $v \succ u$ can damage the security of the key κ_u.

A separating example: Consider a graph $G = (V, E)$ having linear structure, i.e. $V = \{u_0, \ldots, u_{n-1}\}$ with $u_{i+1} \lessdot u_i$ for all i. Let H be a one-way function, which we model as a random oracle. We select S_{u_0} at random from the domain of H and set $\kappa_{u_i} = S_{u_i}$ and $S_{u_{i+1}} = \mathsf{H}(S_{u_i})$ for all i. It is clear how the Gen and Der algorithms should be defined, and that the resulting scheme satisfies the correctness property. It is also easy to see that the scheme is KR-ST-secure in the random oracle model, in the sense of [8]. However, it is also clear that with knowledge of key $\kappa_{u_0} = S_{u_0}$, all keys in the hierarchy can be efficiently determined (including the challenge key κ_u) and hence the scheme is insecure in the S-KR-ST model.

We note that this separation is for key recovery security notions. It is an open problem to construct a separating example for the corresponding key indistinguishability notions.



(a) A poset $(V, \preceq)$ (b) A partition of V into chains

Figure 5.1: Example of a Chain Partition.

5.3 The Chain Partition Construction: A Security Analysis

We begin by reviewing the Chain Partition Construction for key assignment schemes from [48]. Given a partially ordered set $(V, \preceq)$, represented by the directed acyclic graph $P = (V, E)$, we select a particular partition of V into chains $\{C_0, \ldots, C_{w-1}\}$. (Dilworth's Theorem [54] asserts that every partially ordered set $(V, \preceq)$ can be partitioned into w chains, where w is the *width* of V, that is, the cardinality of the largest antichain in V.) The partition need not be unique. The length of C_i is denoted by l_i, for $0 \leq i \leq w-1$. We let $l_{\max}$ denote $\max_i\{l_i\}$. The maximum class of C_i is regarded as the first class in C_i and the minimum class as the last class. Since $\{C_0, \ldots, C_{w-1}\}$ is a partition of V, each $u \in V$ belongs to precisely one chain.

Let $C = u_0 \succcurlyeq \ldots \succcurlyeq u_{l-1}$ be any chain in V. Then any chain of the form $u_j \succcurlyeq \ldots \succcurlyeq u_{l-1}$, $0 < j \leq l-1$ is said to be a *suffix* of C. Now, for any $u \in V$, the set $\downarrow u := \{v \in V : v \preceq u\}$ has non-empty intersection with one or more chains $C_0, \ldots, C_{w-1}$. It is proved in [48] that the intersection of $\downarrow u$ and the chain C_i is a suffix of C_i or the empty set. Following [48], this will enable us to define the private information that should be given to a user in class u.

Since $\{C_0, \ldots, C_{w-1}\}$ is a partition of V into chains, $\{\downarrow u \cap C_0, \ldots, \downarrow u \cap C_{w-1}\}$ is a disjoint collection of chain suffixes. Additionally, the private information for each class in V should be chosen so that the private information for the j-th class of a chain can be used to compute keys for all lower classes in that chain. Hence, we can see that a user in class u should be given the private information for the

maximal classes in the non-empty suffixes $\downarrow u \cap C_0, \ldots, \downarrow u \cap C_{w-1}$. Given $u \in V$, let $\hat{u}_0, \ldots, \hat{u}_{w-1}$ denote these maximal classes, with the convention that $\hat{u}_i = \perp$ if $\downarrow u \cap C_i = \emptyset$. Let u^i_j denote the j-th class in the chain C_i, where $0 \leq j \leq l_i - 1$. Figure 5.1 illustrates a poset $(V, \preceq)$ and a possible partition of V into four chains: C_0, C_1, C_2, C_3. Note that after the poset is partitioned into chains, each $u \in V$ in Figure 5.1(a) is relabelled with u^i_j in Figure 5.1(b), depending on which chain the class is located and on the position of the class in the chain.

The Chain Partition Construction: Let $(V, \preceq)$ be a poset, $P = (V, E)$ the corresponding directed acyclic graph, and k a security parameter. Select a chain partition of V into w chains $C_0, \ldots, C_{w-1}$, so that C_i contains classes $u^i_0, u^i_1, \ldots, u^i_{l_i-1}$, with $u^i_{j+1} \lessdot u^i_j$, $0 \leq j < l_i - 1$. Let $l_{\max}$ denote $\max_i\{l_i\}$. Additionally, let $\mathsf{HKAS}_{\mathsf{chain}} = (\mathsf{Gen}_{\mathsf{chain}}, \mathsf{Der}_{\mathsf{chain}})$ be an HKAS for single chains of length exactly $l_{\max}$. Then the chain partition scheme $\mathsf{HKAS}_{\mathsf{poset}}(\mathsf{HKAS}_{\mathsf{chain}}) = (\mathsf{Gen}_{\mathsf{poset}}, \mathsf{Der}_{\mathsf{poset}})$ (relative to the particular partition selected) is defined as follows.

Algorithm $\mathsf{Gen}_{\mathsf{poset}}(1^k, P)$:

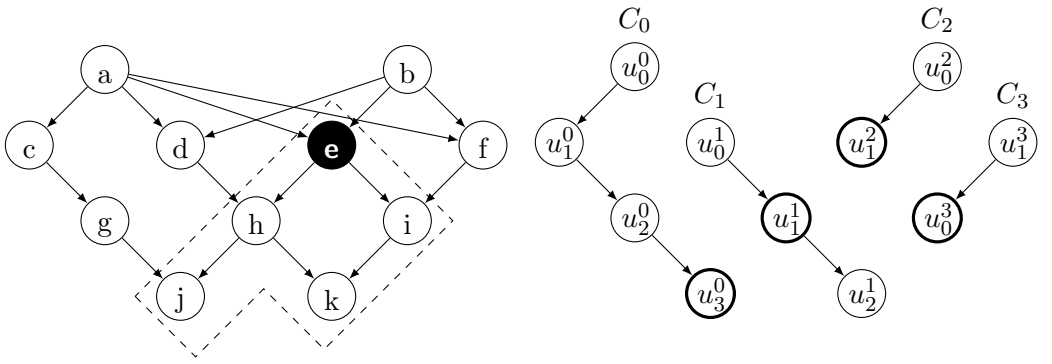
1. For $0 \leq i \leq w-1$, run $\mathsf{Gen}_{\mathsf{chain}}$ on inputs 1^k and a chain of length $l_{\max}$ to obtain (T^i, κ^i, pub^i). Discard the last $l_{\max} - l_i$ elements of T^i and κ^i to obtain the private information and keys for a chain of length l_i. Note that this chain has the same Der algorithm as the starting chain. For ease of notation, we continue to denote the reduced sets by T^i and κ^i, and we write $T^i = \{T_{u^i_0}, \ldots, T_{u^i_{l_i-1}}\}$ and $\kappa^i = \{\kappa_{u^i_0}, \ldots, \kappa_{u^i_{l_i-1}}\}$.

2. For each $u \in V$, define the private information S_u to be $\{T_{\hat{u}_i} : \hat{u}_i \neq \perp, 0 \leq i \leq w-1\}$ and the encryption key κ_u to be $\kappa_u = \kappa_{u^i_j}$, where we assume that u is the j-th class of chain C_i, that is, $u = u^i_j$.

3. Let S and κ be the sets of private information and keys, respectively, in the above construction, and let $pub_{\mathsf{poset}} = (pub^0, \ldots, pub^{w-1})$.

4. Output $(S, \kappa, pub_{\mathsf{poset}})$.

Figure 5.2 illustrates the assignment of private information and encryption keys for an arbitrary poset $(V, \preceq)$. Figure 5.2(a) shows a class e and its descendants



(a) Class e and its descendants

(b) $S_e = T_{u_3^0}, T_{u_1^1}, T_{u_1^2}, T_{u_0^3}$ and $\kappa_e = \kappa_{u_1^2}$

Figure 5.2: The Chain Partition Construction – Assignment of Private Keys and Encryption Keys.

inside a dashed shape. In Figure 5.2(b) we can see that after the poset $(V, \preceq)$ is partitioned into chains, class e is represented by class u_1^2. The highlighted classes $u_3^0, u_1^1, u_1^2, u_0^3$ correspond, respectively, to the maximal classes $\hat{u}_0, \hat{u}_1, \hat{u}_2, \hat{u}_3$ in the suffixes $\downarrow e \cap C_0, \ldots, \downarrow e \cap C_3$. This gives us $S_e = T_{u_3^0}, T_{u_1^1}, T_{u_1^2}, T_{u_0^3}$ and $\kappa_e = \kappa_{u_1^2}$.

Algorithm $\mathsf{Der}_{\mathsf{poset}}(P, u, v, S_u, pub_{\mathsf{poset}})$:

1. Assume that after the chain partition, classes u and v are identified, respectively, as classes u_j^i and u_h^g. For $u \succeq v$, find $\hat{u}_g$, the maximal class in $\downarrow u \cap C_g$. This class is in chain C_g and we denote it by u_r^g, where $0 \leq r \leq h$. Note that, by construction, $T_{u_r^g} \in S_u$.
2. Set $\kappa_v = \kappa_{u_h^g} \leftarrow \mathsf{Der}_{\mathsf{chain}}(C_g, u_r^g, u_h^g, T_{u_r^g}, pub^g)$.
3. Output κ_v.

To illustrate the key derivation process, consider Figure 5.2(b) and suppose we have $S_e = S_{u_1^2}$ and wish to derive the encryption key κ_k for class k, labelled as u_2^1 after the chain partition. As u_2^1 is in chain C_1, we need to find the maximal class in $\downarrow e \cap C_1$, which is $\hat{u}_1 = u_1^1$. Now as u_1^1 is an ascendant class of u_2^1, which is also in chain C_1, and as $T_{u_1^1} \in S_{u_1^2}$ we can use $\mathsf{Der}_{\mathsf{chain}}$ on input $T_{u_1^1}$ to obtain $\kappa_k = \kappa_{u_2^1}$.

Theorem 5.1 (Security of the Chain Partition Construction)**.** *Let Γ be a set of directed acyclic graphs $P = (V, E)$ and l_{max} be the maximum length of the chains in a*

chain partition of V. Let $\mathsf{HKAS}_{\mathsf{chain}}$ be an S-KI-ST-secure scheme for single chains of length l_{max}. Then the scheme $\mathsf{HKAS}_{\mathsf{poset}}(\mathsf{HKAS}_{\mathsf{chain}}) = (\mathsf{Gen}_{\mathsf{poset}}, \mathsf{Der}_{\mathsf{poset}})$ for Γ obtained from the above chain partition construction is also S-KI-ST-secure. More precisely, for every S-KI-ST adversary $\mathcal{A}_{\mathsf{poset}}$ that breaks $\mathsf{HKAS}_{\mathsf{poset}}(\mathsf{HKAS}_{\mathsf{chain}})$ with advantage $\mathrm{Adv}^{\text{S-KI-ST}}_{\mathcal{A}_{\mathsf{poset}},\Gamma}(k)$, there exists an S-KI-ST adversary $\mathcal{A}_{\mathsf{chain}}$ that breaks $\mathsf{HKAS}_{\mathsf{chain}}$ with the same advantage. Moreover, the two adversaries run in roughly the same time.

Proof. Assume $\mathcal{A}_{\mathsf{poset}}$ attacks a class u^i_j of graph $P \in \Gamma$. If $\mathcal{A}_{\mathsf{poset}}$ is able to distinguish between the real key $\kappa_{u^i_j}$ associated with class u^i_j, and a random string having the same length, we show that we can construct an S-KI-ST adversary $\mathcal{A}_{\mathsf{chain}}$ against the scheme $\mathsf{HKAS}_{\mathsf{chain}}$ that, using $\mathcal{A}_{\mathsf{poset}}$ as a black box, is able to distinguish between real or random keys. Algorithm $\mathcal{A}_{\mathsf{chain}}$ plays the S-KI-ST security experiment described in Definition 5.2, receiving as initial input a security parameter 1^k and a chain with $l_{\max}$ classes. Adversary $\mathcal{A}_{\mathsf{chain}}$ simulates the environment of $\mathcal{A}_{\mathsf{poset}}$ in such a way that $\mathcal{A}_{\mathsf{poset}}$'s view is indistinguishable from its view when playing the S-KI-ST security experiment.

Algorithm $\mathcal{A}_{\mathsf{chain}}$:

1. Receive from the S-KI-ST experiment a chain C with $l_{\max}$ classes $v_0, \ldots,$ $v_{l_{\max}-1}$.
2. Pick $P = (V, E) \in \Gamma$ and run $\mathcal{A}_{\mathsf{poset}}$ with input $(1^k, P)$ to get $\mathcal{A}_{\mathsf{poset}}$'s choice of target class u.
3. Generate a chain partition of P containing chains $C_0, \ldots, C_{w-1}$. In this partition, class u is identified as some class u^i_j in some chain C_i of length $l_i \leq l_{\max}$. For $0 \leq t \leq w-1, t \neq i$, run $\mathsf{Gen}_{\mathsf{chain}}$ on inputs 1^k and a chain of length $l_{\max}$ to obtain (S^t, κ^t, pub^t), the set of private information, the set of keys and the public information for that chain. Note that, as in the chain partition construction, these sets can be truncated to obtain the set of private information, the set of keys and the public information for a chain of length exactly l_t. We abuse notation and continue to use (S^t, κ^t, pub^t) to denote this data.
4. Output v_j in chain C as $\mathcal{A}_{\mathsf{chain}}$'s choice of target class. $\mathcal{A}_{\mathsf{chain}}$ now receives

as input the public information, pub, output by $\mathsf{Gen}_{\mathsf{chain}}$, along with private information S_{v_t} for all classes $v_t \prec v_j$ in C, and all keys κ_{v_t} in C such that $v_t \succ v_j$. $\mathcal{A}_{\mathsf{chain}}$ also receives as input a value T which is either the real key κ_{v_j} or a random key of the same length. In what follows, $\mathcal{A}_{\mathsf{chain}}$ will identify the first l_i classes in C with the chain C_i in the chain partition construction.

5. Set $pub_{\mathsf{poset}} = (pub^0, \ldots, pub^{i-1}, pub, pub^{i+1}, \ldots, pub^{w-1})$. Use the private information S_{v_t} for classes $v_t \prec v_j$ in C together with the private information in the sets S^t for $0 \leq t \leq w-1, t \neq i$, to build the set $Corrupt_{P,S,u}$. Use keys κ_{v_t} in C such that $v_t \succ v_j$ and the keys from the sets κ^t to build the set $Keys_{P,u}$.

6. Run $\mathcal{A}_{\mathsf{poset}}$ with inputs $(pub_{\mathsf{poset}}, Corrupt_{P,S,u}, Keys_{P,u}, T)$. It is easy to see that $\mathcal{A}_{\mathsf{chain}}$ has the information required to properly construct the two sets $Corrupt_{P,S,u}$ and $Keys_{P,u}$ in such a way that $\mathcal{A}_{\mathsf{poset}}$'s input here is valid in $\mathcal{A}_{\mathsf{poset}}$'s experiment against the scheme $\mathsf{HKAS}_{\mathsf{poset}}(\mathsf{HKAS}_{\mathsf{chain}})$, and such that T is the real key (resp. the random key) in $\mathcal{A}_{\mathsf{poset}}$'s experiment if and only if T is the real key (resp. the random key) in $\mathcal{A}_{\mathsf{chain}}$'s experiment.

7. When $\mathcal{A}_{\mathsf{poset}}$ outputs a bit, output the same bit.

Now as $\mathcal{A}_{\mathsf{chain}}$'s simulation is perfect, we see that the advantage of $\mathcal{A}_{\mathsf{chain}}$ in winning its S-KI-ST indistinguishability game for the chain C of length $l_{\max}$ is the same as the advantage of $\mathcal{A}_{\mathsf{poset}}$ in playing the S-KI-ST indistinguishability game against $\mathsf{HKAS}_{\mathsf{poset}}(\mathsf{HKAS}_{\mathsf{chain}})$. The theorem now follows. □

Note that, in the above theorem, $\mathsf{HKAS}_{\mathsf{chain}}$ need only be an S-KI-ST-secure scheme for chains of length exactly $l_{\max}$. Because of the truncation trick, this is equivalent to $\mathsf{HKAS}_{\mathsf{chain}}$ being an S-KI-ST-secure scheme for the set of graphs consisting of chains of lengths up to $l_{\max}$.

Remark: We stress that Theorem 5.1 and its security proof can be slightly modified for different HKAS security notions, such as S-KR-ST-security, KI-security and KR-security.

5.4 Srongly KI-Secure Constructions

In this section we construct two very simple and efficient S-KI-ST-secure key assignment schemes for totally ordered hierarchical access structures (chains) of arbitrary depth. The first construction, shown in Section 5.4.1, uses a pseudorandom function, while the second, shown in Section 5.4.2, uses a forward-secure pseudorandom generator. General key assignment schemes for arbitrary posets can be obtained by combining our constructions with the result of Section 5.3.

5.4.1 A PRF-based HKAS for Totally Ordered Hierarchies

Here we use PRFs to construct an S-KI-ST-secure hierarchical key assignment scheme for totally ordered hierarchies, $\mathsf{HKAS}_{\mathsf{PRF}} = (\mathsf{Gen}_{\mathsf{PRF}}, \mathsf{Der}_{\mathsf{PRF}})$.

Let Γ be the family of graphs corresponding to totally ordered hierarchies, and let $G = (V, E) \in \Gamma$ be a graph, where $V = \{u_0, \ldots, u_{n-1}\}$ for some n, and $u_{i+1} \lessdot u_i$ for all i. For better exposition, we abuse notation writing S_i and κ_i (instead of S_{u_i} and κ_{u_i}) for the private information and key assigned to each security class $u_i \in V$. (We can do this because we are in the chain setting.)

Let k be a security parameter and let $\mathsf{F} : \{0,1\}^k \times \mathcal{D} \to \{0,1\}^k$ be a PRF (see Section 2.3.3). Let c_0 and c_1 be two different elements in $\mathcal{D}$. The $\mathsf{Gen}_{\mathsf{PRF}}$ and $\mathsf{Der}_{\mathsf{PRF}}$ algorithms are defined in Figure 5.3. Observe that computing key κ_j from secret information S_i requires exactly $j - i + 1$ evaluations of the underlying PRF.

Note that we use special PRFs where $\mathcal{K} = \mathcal{R}$ and $\mathcal{D}$ is any set.[1] For concreteness, we propose to deploy the (hash-based) HMAC primitive [20] as a PRF (see also analysis in [55]). In addition, it might be possible to find suitable constructions based on number-theoretic assumptions, e.g. derived from the PRF obtained by converting the BBS PRG [26] into a PRF via the Goldreich-Goldwasser-Micali (GGM) construction [76].

[1] We remark that this special restriction applies not only to our PRF-based HKAS, but also to other PRF-based key assignment schemes; Attallah *et al.* [8], for example, provide some PRF-based constructions which also require similar restrictions on the keyspace and range of the PRFs.

Algorithm $\mathsf{Gen}_{\mathsf{PRF}}(1^k, G)$

$S_0 \leftarrow \{0,1\}^k$, $\kappa_0 \leftarrow \mathsf{F}_{S_0}(c_1)$

For each class $u_i \in V\ (i > 0)$, set $S_i \leftarrow \mathsf{F}_{S_{i-1}}(c_0), \kappa_i \leftarrow \mathsf{F}_{S_i}(c_1)$

$S \leftarrow (S_0, \ldots, S_{n-1}), \kappa \leftarrow (\kappa_0, \ldots, \kappa_{n-1}), pub \leftarrow \emptyset$

Output (S, κ, pub)

Algorithm $\mathsf{Der}_{\mathsf{PRF}}(G, u_i, u_j, S_i, pub)$ (note that we may assume $j \geq i$)

If $i = j$, return $\kappa_j = \mathsf{F}_{S_i}(c_1)$

For $h = i + 1$ to j

$S_h \leftarrow \mathsf{F}_{S_{h-1}}(c_0)$

Return $\kappa_j = \mathsf{F}_{S_j}(c_1)$

Figure 5.3: The Hierarchical Key Assignment Scheme $\mathsf{HKAS}_{\mathsf{PRF}}$.

Theorem 5.2 (Security of the PRF-based HKAS)**.** *Assume* F *is a PRF and* Γ *is a family of graphs corresponding to totally ordered hierarchies. Then* $\mathsf{HKAS}_{\mathsf{PRF}}$ *is S-KI-ST-secure for any graph* $G \in \Gamma$*.*

Proof. Fix any totally ordered graph $G = (V, E) \in \Gamma$. Proving the theorem amounts to showing that the only way to break the S-KI-ST-security of $\mathsf{HKAS}_{\mathsf{PRF}}$ is by breaking the pseudorandom function F. To this end, we need to show how to turn an S-KI-ST adversary $\mathcal{A}$ attacking $\mathsf{HKAS}_{\mathsf{PRF}}$ into an adversary $\mathcal{A}_{\mathsf{F}}$ attacking F. Assume $\mathcal{A}$ attacks a class $u_i \in V$. We define a sequence of computationally indistinguishable games $G_0, G_1, \ldots, G_{i+1}$. We start with G_0, defined to be the original experiment $\mathrm{Exp}^{\text{S-KI-ST},\,d}_{\mathcal{A},\Gamma}(1^k)$ as described in Definition 5.2. Then we make successive transitions until we reach G_{i+1}, for which $\mathcal{A}$'s probability of success in outputting the correct bit $d' = d$, for $d \leftarrow \{0,1\}$, is only $1/2$.

Let Succ_ι be the event that $\mathcal{A}$ is successful in Game G_ι.

Game G_0. Let G_0 be the original experiment $\mathrm{Exp}^{\text{S-KI-ST},\,d}_{\mathcal{A},\Gamma}(1^k)$ as described in Definition 5.2.

Game G_ι $(1 \leq \iota \leq i + 1)$. This game is identical to $G_{\iota-1}$, except that here the assignment of private information and keys is modified in such a way that key $\kappa_{\iota-1}$ and private information S_ι are substituted with values randomly selected from $\{0,1\}^k$.

This modification amounts to substituting the occurrences of $\mathsf{F}_{S_{\iota-1}}$ in $G_{\iota-1}$ with a truly random function, which is warranted by the security of the PRF, as we will see in Lemma 5.1. Hence,

$$|\Pr[\text{Succ}_\iota] - \Pr[\text{Succ}_{\iota-1}]| \leq \epsilon_{\text{PRF}}(k) \ ,$$

where $\epsilon_{\text{PRF}}(k)$ is a bound on the advantage $\text{Adv}^{\text{PRF}}_{\mathcal{A}_\mathsf{F},\mathsf{F}}(k)$ for any polynomial-time adversary $\mathcal{A}_\mathsf{F}$.

Now, we see that in G_{i+1} the adversary's view is independent of bit d; in both cases it gets as challenge a random value in $\{0,1\}^k$. Thus,

$$\Pr[\text{Succ}_{i+1}] = 1/2 \ .$$

Therefore, we have

$$\text{Adv}^{\text{S-KI-ST}}_{\mathcal{A},\Gamma}(k) = 2\,|\Pr[\text{Succ}_0] - 1/2| \leq 2(i+1)\epsilon_{\text{PRF}}(k) \ .$$

By assumption $\epsilon_{\text{PRF}}(k)$ is negligible, concluding the proof. □

Lemma 5.1. $|\Pr[Succ_\iota] - \Pr[Succ_{\iota-1}]| \leq \epsilon_{\text{PRF}}(k)$.

Proof. Assume we have an S-KI-ST adversary $\mathcal{A}$ against $\mathsf{HKAS}_{\mathsf{PRF}}$ that attacks a class u_i and is able to distinguish between games $G_{\iota-1}$ and G_ι. We describe below how to construct an algorithm $\mathcal{A}_\mathsf{F}$ that, using $\mathcal{A}$ as a black-box, is able to distinguish between pseudorandom and truly random functions.

Algorithm $\mathcal{A}_\mathsf{F}$ plays the PRF experiment described in Definition 2.28 and is thus given access to a function f_β that is either an instance of a pseudorandom function $\mathsf{F}_\kappa : \mathcal{D} \to \{0,1\}^k$, keyed with a random key $\kappa \leftarrow \{0,1\}^k$ (if $\beta = 1$), or a truly random function into $\{0,1\}^k$ (if $\beta = 0$). In order to use algorithm $\mathcal{A}$, $\mathcal{A}_\mathsf{F}$ simulates the environment of $\mathcal{A}$ in such a way that interpolates between games $G_{\iota-1}$ and G_ι. This means that if $\mathcal{A}_\mathsf{F}$ is interacting with a pseudorandom function, then the simulation proceeds as in $G_{\iota-1}$. Otherwise, if $\mathcal{A}_\mathsf{F}$ is interacting with a random function, then the simulation proceeds as in G_ι. More formally, algorithm $\mathcal{A}_\mathsf{F}$ works as follows.

Algorithm $\mathcal{A}_\mathsf{F}$:

1. Run $\mathcal{A}$ with input $(1^k, G)$ to get $\mathcal{A}$'s choice of target class u_i.

2. Set up the access hierarchy for graph G by running $\mathsf{Gen}_{\mathsf{PRF}}(1^k, G)$ with the following modifications:

   (a) Private information S_ι and key $\kappa_{\iota-1}$ are computed via oracle f_β as follows:

   $$S_\iota \leftarrow f_\beta(c_0),\ \kappa_{\iota-1} \leftarrow f_\beta(c_1).$$

   (b) For all $\iota' < \iota$, set $S_{\iota'} \leftarrow \{0,1\}^k$.

   (c) For all $\iota' < \iota - 1$, set $\kappa_{\iota'} \leftarrow \{0,1\}^k$.

   So far, this is almost equivalent to the actions of $G_{\iota-1}$ when $\beta = 1$ and equivalent to G_ι when $\beta = 0$. The only difference is that, in case $\beta = 1$ (i.e. $\mathcal{A}_\mathsf{F}$ has oracle acess to a PRF F_κ for unknown $\kappa \leftarrow \{0,1\}^k$), the private information $S_{\iota-1}$ is not equal to κ. This is completely indistinguishable from $\mathcal{A}$'s view since both $S_{\iota-1}$ and κ are uniformly chosen from $\{0,1\}^k$ and $\mathcal{A}$ is not allowed to ask for private information $S_{i'}$, $i' < i+1$. Note that as $\iota \leq i+1$, $\mathcal{A}$ is not allowed to ask for $S_{\iota'}$, $\iota' < \iota$.

3. Pick a random bit d. If $d = 1$, $\mathcal{A}_\mathsf{F}$ sets $\mathcal{A}$'s challenge, denoted here by T, to be the real key κ_i. If $d = 0$, $\mathcal{A}_\mathsf{F}$ sets T to be a random key of the same length as κ_i.

4. Run $\mathcal{A}$ with inputs $(pub, Corrupt_{G,S,u_i}, Keys_{G,u_i}, T)$, where $Corrupt_{G,S,u_i} = \{S_{i+1}, \ldots, S_{n-1}\}$ and $Keys_{G,u_i} = \{\kappa_0, \ldots, \kappa_{i-1}\}$, to obtain a bit d'. (Note that according to $\mathcal{A}_\mathsf{F}$'s set-up of the access hierarchy, it can compute all this information.) Here d' is $\mathcal{A}$'s guess as to whether it was given the real key associated with class u_i or a random string having the same length.

5. If $d' = d$, output $\beta' = 1$, guessing for a pseudorandom function; otherwise, output $\beta' = 0$, guessing for a truly random function.

Now we have

$$\begin{aligned}
\epsilon_{\mathrm{PRF}}(k) &\geq \mathrm{Adv}^{\mathrm{PRF}}_{\mathcal{A}_\mathsf{F},\mathsf{F}}(k) \\
&= \left|\Pr[\mathrm{Exp}^{\mathrm{PRF},1}_{\mathcal{A}_\mathsf{F},\mathsf{F}}(1^k) = 1] - \Pr[\mathrm{Exp}^{\mathrm{PRF},0}_{\mathcal{A}_\mathsf{F},\mathsf{F}}(1^k) = 1]\right| \\
&= \left|\Pr[1 \leftarrow \mathcal{A}_\mathsf{F} \mid \beta = 1] - \Pr[1 \leftarrow \mathcal{A}_\mathsf{F} \mid \beta = 0]\right| \\
&= \left|\Pr[d' = d \mid \beta = 1] - \Pr[d' = d \mid \beta = 0]\right| \\
&= \left|\Pr[\mathrm{Succ}_{\iota-1}] - \Pr[\mathrm{Succ}_\iota]\right|.
\end{aligned}$$

□

Algorithm $\mathsf{Gen}_{\mathsf{FS\text{-}PRG}}(1^k, G)$

$params \leftarrow \mathsf{G}_{\mathsf{FS}}.\mathsf{Setup}(1^k), S_0 \leftarrow \mathsf{G}_{\mathsf{FS}}.\mathsf{Key}(params)$

For all $0 \leq i < n$, compute $(S_{i+1}, \kappa_i) \leftarrow \mathsf{G}_{\mathsf{FS}}.\mathsf{Next}(S_i)$

$S \leftarrow (S_0, \ldots, S_{n-1}), \kappa \leftarrow (\kappa_0, \ldots, \kappa_{n-1}), pub \leftarrow \emptyset$

Output (S, κ, pub)

Algorithm $\mathsf{Der}_{\mathsf{FS\text{-}PRG}}(G, u_i, u_j, S_i, pub)$ (note that we may assume $j \geq i$)

For $h = i$ to j

$(S_{h+1}, \kappa_h) \leftarrow \mathsf{G}_{\mathsf{FS}}.\mathsf{Next}(S_h)$

Return κ_j

Figure 5.4: The Hierarchical Key Assignment Scheme $\mathsf{HKAS}_{\mathsf{FS\text{-}PRG}}$.

5.4.2 An FS-PRG-based HKAS for Totally Ordered Hierarchies

Here, building on generic FS-PRGs, we construct an S-KI-ST secure hierarchical key assignment scheme for totally ordered access graphs, $\mathsf{HKAS}_{\mathsf{FS\text{-}PRG}} = (\mathsf{Gen}_{\mathsf{FS\text{-}PRG}}, \mathsf{Der}_{\mathsf{FS\text{-}PRG}})$. When combined with the result from Section 5.3, our construction widely generalizes our construction for arbitrary posets from [62], which implicitly exploits the property of forward security of the BBS pseudorandom generator (see Construction 2.1 in Section 2.3.4). As our construction generically builds on FS-PRGs, it is amenable to the efficiency gain obtained by replacing the BBS-based FS-PRG by, for instance, an HMAC-based one. In the following we describe our scheme for totally ordered hierarchies $\mathsf{HKAS}_{\mathsf{FS\text{-}PRG}}$.

Let Γ be the family of graphs corresponding to totally ordered hierarchies, and let $G = (V, E) \in \Gamma$ be a graph, where $V = \{u_0, \ldots, u_{n-1}\}$ for some n, and $u_{i+1} \lessdot u_i$ for all i. As in Section 5.4.1, we write S_i for private information S_{u_i}, and κ_i for key κ_{u_i}.

Let k be a security parameter, and let $\mathsf{G}_{\mathsf{FS}} = (\mathsf{G}_{\mathsf{FS}}.\mathsf{Setup}, \mathsf{G}_{\mathsf{FS}}.\mathsf{Key}, \mathsf{G}_{\mathsf{FS}}.\mathsf{Next})$ be an FS-PRG with output blocks of length $p(k)$ (see Definition 2.31). The $\mathsf{Gen}_{\mathsf{FS\text{-}PRG}}$ and $\mathsf{Der}_{\mathsf{FS\text{-}PRG}}$ algorithms are defined in Figure 5.4. Note that we identify the FS-PRG's state St_i with the private information S_i stored for class u_i, while key κ_i is set to the FS-PRG's output Out_{i+1}.

Theorem 5.3 (Security of the FS-PRG-based HKAS)**.** *Assume* G_{FS} *is an FS-PRG and* Γ *is a family of graphs corresponding to totally ordered hierarchies. Then*

$\mathsf{HKAS}_{\mathsf{FS\text{-}PRG}}$ *is S-KI-ST-secure for any graph* $G \in \Gamma$.

Proof. Fix any totally ordered graph $G = (V, E) \in \Gamma$. This proof amounts to showing that for every S-KI-ST adversary $\mathcal{A}$ that breaks our FS-PRG-based scheme $\mathsf{HKAS}_{\mathsf{FS\text{-}PRG}}$ with advantage $\mathrm{Adv}^{\text{S-KI-ST}}_{\mathcal{A},\Gamma}(k)$, there exists an algorithm $\mathcal{D}$ that breaks the forward-security of G_{FS} with advantage $\mathrm{Adv}^{\text{FS-PRG}}_{\mathcal{D},\mathsf{G}_{\mathsf{FS}}}(k) \geq \mathrm{Adv}^{\text{S-KI-ST}}_{\mathcal{A},\Gamma}(k)/2$. Moreover, $\mathcal{D}$ has roughly the same running time as $\mathcal{A}$. Assume we have an S-KI-ST adversary $\mathcal{A}$ that attacks a class u_i and is able to distinguish between the real key κ_i assigned to that class, and a random string of the same length. Then, we can construct an algorithm $\mathcal{D}$ that, using $\mathcal{A}$ as a black box, is able to tell apart output blocks generated by G_{FS}, up to some iteration t, from a sequence of purely random blocks.

Algorithm $\mathcal{D}$ initially outputs an iteration number, say t. Then it is given access to a sequence of t output blocks $Out = Out_1, \ldots, Out_t$, and the current state, St_t, of the FS-PRG. $\mathcal{D}$ has to distinguish if Out is a sequence of the first t output blocks of G_{FS} or a sequence of random blocks each of length $p(k)$. (See the security experiment of Definition 2.31.) Given all that information, $\mathcal{D}$ simulates the environment of $\mathcal{A}$ in a way that $\mathcal{A}$'s view is indistinguishable from the S-KI-ST experiment of Definition 5.2. We want to prove that

$$\begin{aligned} \mathrm{Adv}^{\text{S-KI-ST}}_{\mathcal{A},\Gamma}(k) &= \left|\Pr[\mathrm{Exp}^{\text{S-KI-ST},1}_{\mathcal{A},\Gamma}(1^k) = 1] - \Pr[\mathrm{Exp}^{\text{S-KI-ST},0}_{\mathcal{A},\Gamma}(1^k) = 1]\right| \\ &\leq 2\,\epsilon_{\mathrm{FS-PRG}}(k), \end{aligned}$$

where $\epsilon_{\mathrm{FS-PRG}}$ is an upper bound on the advantage $\mathrm{Adv}^{\text{FS-PRG}}_{\mathcal{D},\mathsf{G}_{\mathsf{FS}}}(k)$ for any polynomial-time distinguisher $\mathcal{D}$. We define a sequence of games and analyse it.

Game G_0. Define Game G_0 to be identical to $\mathrm{Exp}^{\text{S-KI-ST,d}}_{\mathcal{A},\Gamma}(1^k)$ for $d = 0$. In particular, the challenge key T is chosen to be random in $\{0,1\}^{p(k)}$.

Game G_1. This game is like Game G_0, except that all elements in $Keys_{G,u}$ are replaced by random strings of length $p(k)$. Challenge key T remains as before, that is, it is a random value in $\{0,1\}^{p(k)}$.

Game G_2. This game is identical to $\mathrm{Exp}^{\text{S-KI-ST,d}}_{\mathcal{A},\Gamma}(1^k)$ for $d = 1$. In particular, the challenge key T is the real key κ_i assigned to class u_i, as computed via $\mathsf{Gen}_{\mathsf{FS\text{-}PRG}}$.

In the analysis, let $\mathcal{A}(G_\iota)$ denote adversary $\mathcal{A}$ playing game G_ι. First we show that $|\Pr[\mathcal{A}(G_0)=1] - \Pr[\mathcal{A}(G_1)=1]| \leq \epsilon_{\mathrm{FS-PRG}}(k)$. Consider the following distinguisher $\mathcal{D}$ against the FS-PRG $\mathsf{G_{FS}}$.

$\mathcal{D}$ runs $\mathcal{A}$ with input $(1^k, G)$ to get u_i, the class that $\mathcal{A}$ is aiming to attack. $\mathcal{D}$ chooses an integer $t = i$ and sends it to its own challenger. It receives St_i and $Out = Out_1, \ldots, Out_i$, where the latter is either the first i output blocks generated by $\mathsf{G_{FS}}.\mathsf{Next}$, or a collection of random strings in $\{0,1\}^{p(k)}$. $\mathcal{D}$ sets $Keys_{G,u_i} = Out_1, \ldots, Out_i$ and uses St_i to compute the private information collected in $Corrupt_{G,S,u_i}$. Then $\mathcal{D}$ sets $Corrupt_{G,S,u_i} = \{St_{i+1}, \ldots, St_{n-1}\}$, $pub \leftarrow \emptyset$, $T \leftarrow \{0,1\}^{p(k)}$ and, finally runs $\mathcal{A}$ on input $(pub, Keys_{G,u_i}, Corrupt_{G,S,u_i}, T)$. Whenever $\mathcal{D}$ receives a bit d' from $\mathcal{A}$, $\mathcal{D}$ forwards the same bit to its own challenger. We see that if Out is the sequence of first i output blocks generated by $\mathsf{G_{FS}}.\mathsf{Next}$, then $\mathcal{A}$'s view is exactly as in game G_0. On the other hand, if the values are random strings, then $\mathcal{A}$'s view is exactly as in game G_1. Now we have

$$\begin{aligned}
\epsilon_{\mathrm{FS-PRG}}(k) &\geq \mathrm{Adv}^{\text{FS-PRG}}_{\mathcal{D},\mathsf{G_{FS}}}(k) \\
&= \left|\Pr\left[\mathrm{Exp}^{\text{FS-PRG},1}_{\mathcal{D},\mathsf{G_{FS}}}(1^k)=1\right] - \Pr\left[\mathrm{Exp}^{\text{FS-PRG},0}_{\mathcal{D},\mathsf{G_{FS}}}(1^k)=1\right]\right| \\
&= \left|\Pr[\mathcal{A}(G_0)=1] - \Pr[\mathcal{A}(G_1)=1]\right|.
\end{aligned}$$

Next we bound $|\Pr[\mathcal{A}(G_1)=1] - \Pr[\mathcal{A}(G_2)=1]|$. The reduction is similar to the one above, the difference is that this time the FS-PRG distinguisher $\mathcal{D}$ specifies $t = i+1$ instead of $t = i$, and assigns $\kappa_j \leftarrow Out_{j+1}$ for all $0 \leq j \leq i$. This means that not only the keys in $Keys_{G,u_i}$ are taken from Out, but also the 'challenge key' κ_i. Notice that here, $\mathcal{D}$ has access to the value St_{i+1} and thus is able to compute the set of private information $Corrupt_{G,S,u_i} = \{St_{i+1}, \ldots, St_{n-1}\}$. Similarly to above, this establishes $|\Pr[\mathcal{A}(G_1)=1] - \Pr[\mathcal{A}(G_2)=1]| \leq \epsilon_{\mathrm{FS-PRG}}(k)$.

All in all, this proves that

$$\begin{aligned}
\mathrm{Adv}^{\text{S-KI-ST}}_{\mathcal{A},\Gamma}(k) &= \left|\Pr[\mathrm{Exp}^{\text{S-KI-ST},1}_{\mathcal{A},\Gamma}(1^k)=1] - \Pr[\mathrm{Exp}^{\text{S-KI-ST},0}_{\mathcal{A},\Gamma}(1^k)=1]\right| \\
&= |\Pr[\mathcal{A}(G_2)=1] - \Pr[\mathcal{A}(G_0)=1]| \\
&\leq 2\,\epsilon_{\mathrm{FS-PRG}}(k).
\end{aligned}$$

The theorem now follows. □

5.5 Comparison with Previous Schemes

In this section, we compare instantiations of our constructions with other provably secure schemes from the literature. Note that all these previous schemes have proofs only in weaker security models than our strong security models. (However, in some cases, these schemes can also be proven secure in our strong security models.)

In [8, 9], Atallah *et al.* proposed a first construction based on pseudorandom functions, which they prove to be KR-secure; and a second construction which they prove to be KI-secure, but which requires the additional use of a symmetric encryption scheme secure under chosen plaintext attack (SE-CPA). In both constructions the private information of a class consists of a single symmetric key associated with that class. In the first construction, the amount of public information grows with the number of edges in the directed acyclic graph (each edge has an associated PRF output). In the second construction, the amount of public information grows with the number of classes (each class has an associated ciphertext). In both constructions, key derivation uses only symmetric operations and its cost grows linearly with the number of levels between the classes.

De Santis *et al.* [123] proposed a construction which is based on symmetric encryption schemes only, achieves KI-security and is simpler than the KI-secure scheme proposed in [8]. The construction uses only a chosen plaintext secure symmetric encryption scheme and the required private key storage is small at one key per class. The amount of public storage needed grows linearly with the number of classes and the number of edges in the graph. Key derivation requires roughly h symmetric decryptions, where h is the number of levels between the classes.

Ateniese *et al.* [10, 11] proposed two different constructions for time-bound key assignment schemes, which achieve KI-security. Their first construction is based on symmetric encryption schemes and the second makes use of bilinear maps. The security of the latter construction is based on the Decisional Bilinear Diffie-Hellman (DBDH) assumption. The advantage of these constructions is that they provide very efficient procedures for key derivation, requiring only one decryption or one pairing evaluation, no matter the number of levels between the classes. However, the public information for a scheme obtained from the first construction can be very large since

it depends not only on the number of classes, but also on the number of time periods. The downside of the second construction is that the private information could be as large as the number of time periods (and the public information is already very large). These constructions are not directly comparable to normal (i.e. non-time-bound) schemes, but we include them in the comparison to get an idea of their efficiency compared to normal schemes.

D'Arco *et al.* [50] proposed a generic construction yielding a key assignment scheme offering KI-security, using as components a KR-secure scheme and the Goldreich-Levin hard-core bit (GL bit) [74]. They instantiated their construction with an Akl-Taylor scheme [5], which they proved to be KR-secure under the RSA assumption, therefore achieving KI-security under the same assumption. Their construction can also be instantiated, for example, with the KR-secure scheme from Atallah *et al.* [8], yielding a KI-secure scheme based on PRFs. However, their construction involves a significant "blow up" when going from KR-security to KI-security, since it requires the use of a KR-secure scheme for a poset having ℓ classes for each class in the poset for the final KI-secure scheme, where ℓ is the length of keys. Typically, we would desire $\ell = 128$ bits in applications. Their construction also consumes a large amount of public storage, requiring roughly ℓ times as many public values as in the starting KR-secure scheme. Key derivation also involves an increase by a factor of ℓ relative to the starting scheme.

Our constructions provide stronger security guarantees (i.e. key indistinguishability in our *strengthened* model) than all the above schemes and indeed all other hierarchical key assignment schemes in the literature. Our constructions provide schemes having a trade-off between storage of private information and efficiency of key derivation, depending on how the poset is partitioned into chains. The overall efficiency of key derivation is bounded by the length of the longest chain in the partition, and the amount of private information depends on w, the poset width (which is equal to the number of chains in the partition). Moreover, due to the cryptographic components used in our constructions (PRFs and FS-PRGs), key derivation is relatively efficient, growing linearly in h, the number of levels between classes. In addition, our constructions require no public storage.

Table 5.1 gives us a comparison of our schemes and other provably secure schemes

Scheme	Private storage	Public storage	Key derivation	Type of security	Security based on
Atallah *et al.* [8, 9]	k	$k(\|E\| + \|V\|)$	$(c_H + c_{XOR})h$	KR	PRF
	k	$k_1(2\|E\| + \|V\|)$	$c_H(h+1) + c_D h$	KI	PRF+SE-CPA
De Santis *et al.* [123]	k	$k_1(\|E\| + \|V\|)$	$c_D(h+1)$	KI	SE-CPA
Ateniese *et al.* [10, 11]	k	$k_1\mathcal{O}(\|V\|^2 \cdot \|T\|^3)$	c_D	KI	SE-CPA
	$k_4\mathcal{O}(\|T\|)$	$k_3\mathcal{O}(\|V\|^2)$	c_P	KI	DBDH
D'Arco *et al.* [50] (+[5])	k_2	$k_2(\|V\|(1+\ell)+2)$	$c_E \cdot \ell$	KI	Random exp. RSA
Ours (PRF-based)	$w \cdot k$	0	$c_{\text{PRF}}(h+1)$	S-KI	PRF
Ours (FS-PRG-based)	$w \cdot k_2$	0	$c_{\text{FS-PRG}}(h+1)$	S-KI	FS-PRG

Table 5.1: Comparison with previous schemes.

in the literature. Private storage is measured per class in the access graph, public storage is measured for the entire hierarchy, and key derivation shows the maximum amount of computation that is needed to traverse h levels down in the hierarchy. We also include the type of security that the scheme achieves, key recovery (KR), key indistinguishability (KI) or our strengthened version, S-KI, and the basic components of the scheme. In the table, k is a security parameter that corresponds to the size of the keys for a PRF or for a symmetric encryption scheme $\mathcal{E}$; k_1 represents the size of ciphertexts for a semantically secure symmetric encryption scheme; k_2 is a security parameter for a pseudorandom generator; k_3 represents the size of pairing-friendly group elements (which is typically a little larger than twice the security parameter, e.g. 171 bits at the 80-bit security level); k_4 is the size of the order q of a pairing-friendly group (which can usually be taken as twice the security parameter); c_H denotes the computation required to compute the output of a hash function; c_D is the computation needed for a symmetric key decryption; c_P is the computation needed for one pairing evaluation; c_E is the cost of exponentiation modulo an integer N of size k_2; c_{PRF} is the computation needed to compute the output of a PRF; $c_{\text{FS-PRG}}$ is the computation needed to compute the output of an FS-PRG; ℓ is the bit-size of the scheme's keys; and w is the width of a poset. Finally, $|T|$ represents the number of distinct time periods in the time-bound schemes of [10, 11].

Chapter 6

Concluding Remarks

In this chapter, we highlight the contributions of this thesis to the area of cryptography and discuss a number of possible research directions.

In this thesis we have addressed two distinct topics. In Part I we exploited the non-interactive key exchange (NIKE) primitive in two settings. First we systematically studied NIKE in the public key setting. We argued that NIKE has received scant attention since the 1976 paper by Diffie and Hellman [53] – prior to our work, the only existing formal security model for NIKE was the one by Cash, Kiltz and Shoup (CKS) [40], who analysed the Diffie-Hellman NIKE scheme, as well as a variant of it, in the random oracle model. We therefore provided different security models for NIKE and explored the relationships between them and also the CKS security model; we focused on the challenging scenario where an adversary can register arbitrary public keys into the system, what we call the *dishonest key registration* (DKR) model. Having proven that those models are all polynomially equivalent, we were then able to construct and analyse NIKE schemes in the simplest security model, which we called the CKS-light model. We constructed a pairing-based scheme with security in the standard model, and a variant of the Diffie-Hellman scheme with security in the random oracle model, under the factoring assumption. Furthermore, we also showed that a secure NIKE scheme can be converted into an IND-CCA PKE scheme, thus illustrating the fundamental nature of NIKE in public key cryptography.

We then considered NIKE in the identity-based setting (ID-NIKE). Using multilinear maps, we constructed the *first ever* ID-NIKE scheme with security in the *standard model*. Not only that, we also constructed the *first ever* fully-secure hierarchical ID-

NIKE (H-ID-NIKE) scheme with security either in the random oracle model or in the standard model. Although not very efficient, our constructions demonstrate that it is possible to construct (H-)ID-NIKE schemes which are secure in the standard model, moving away from the ROM-secure SOK scheme [122].

In Part II of this thesis, we first proposed enhanced security notions for hierarchical key assignment schemes (HKASs). Our notions, which we call strong key-indistinguishability (S-KI) and strong key recovery (S-KR), are, respectively, strengthened versions of the KI and KR security notions of Atallah *et al.* [8]. They provide the adversary against an HKAS with the additional capability of obtaining keys for classes above the target class. We then provided a general security analysis for the chain partition construction of Crampton *et al.* [48]; we proved that the security of an HKAS for arbitrary posets, built using the chain partition construction along with an HKAS for chains, is equivalent to the security of the HKAS for chains. We gave simple and efficient constructions for HKASs and proved them to be secure in our S-KI security notion. Our constructions offer an attractive trade-off between storage requirements and efficiency of key derivation.

6.1 Directions for Future Research

The work presented in this thesis has identified several research directions. We discuss these and a few additional ones below.

1. **Factoring-based NIKE scheme in the standard model.** In Chapter 3, we provided a pairing-based NIKE scheme and proved it to be secure in the standard model, in the DKR setting, under the DBDH-2 assumption. An apparently difficult open problem is to construct a NIKE scheme which is secure under the *factoring assumption* in the standard model, in the DKR setting; as far as we are aware, the only NIKE schemes with security under the factoring assumption in the DKR setting use the random oracle model. The main obstacle for us to solve this goal is to be able to non-interactively check the validity of arbitrary public keys registered by the adversary.

2. **Multi-user NIKE.** In 2000 Joux [93] proposed a 3-user NIKE scheme which

unfortunately suffers from the same security issue as the original Diffie-Hellman NIKE scheme (as discussed in Chapter 1); its hashed version, however, is secure in the random oracle model. For the case of $n \geq 3$, Boneh and Silverberg [33] showed how to generalize Joux's protocol to an n-user NIKE scheme if multilinear maps existed. Garg, Gentry and Halevi [72] in turn, using their candidate multilinear maps, adapted the construction of [33], providing an instantiation of an n-user NIKE scheme. However, like Joux's protocol, the constructions of [33] and [72] are not secure in the setting where the adversary is allowed to arbitrarily register public keys against users of its choice, i.e. they are not secure in our DKR setting. An interesting open problem is to construct an n-user NIKE scheme (for any constant $n \geq 3$) which is secure in the *standard model*, in the DKR setting. However, a requirement, as a first step towards this goal, would be to extend our security models for (2-user) NIKE to the n-user setting.

3. **Multi-user (H-)ID-NIKE.** As we mentioned in Chapter 4, it would be very interesting to formally extend our work on (H-)ID-NIKE to the setting where shared keys can be computed by groups of users instead of only two users. Research in this direction would yield the first secure multi-user (H-)ID-NIKE schemes in the literature.

4. **Multi-TA (H-)ID-NIKE.** Again, as we discussed in Chapter 4, our work on (H-)ID-NIKE can be extended to the setting with multiple independent TAs. This is a more practical scenario since, for example, a pair of users who obtained keys from independent TAs may still want to compute a shared key between themselves. It would also be very interesting to go further and construct a multi-TA, multi-user (H-)ID-NIKE scheme.

5. **Multilinear map-free (H-)ID-NIKE in the standard model.** As mentioned earlier, due to the use of multilinear maps, our (H-)ID-NIKE constructions are not very efficient. In order for (H-)ID-NIKE schemes to be of more practical use, we encourage the search for such constructions which are secure in our security models, but do not require the use of multilinear maps.

6. **Construction of other cryptographic primitives from NIKE or ID-NIKE.** In this thesis we demonstrated that the NIKE primitive is so fundamental that it can even be used to construct IND-CCA PKE schemes. Also,

in [112], Paterson and Srinivasan showed that certain ID-NIKE schemes can be converted into IBE schemes. We propose to further investigate the relationships between (ID-)NIKE schemes and other cryptographic primitives.

7. **"Tree" partition construction for HKAS.** The chain partition construction of Crampton *et al.* [48] that we analysed in Chapter 5 provides an attractive trade-off between storage requirement and efficiency of key derivation. It would be interesting to formally introduce and analyse a construction in which instead of partitioning the access graph into a collection of chains, we partition it into a collection of trees. This construction would benefit from smaller storage requirement and more efficient key derivation. As an example, one could use PRFs to efficiently build schemes for trees and then use the tree partition construction to generalize these schemes to arbitrary hierarchies.

8. **Separating example for HKAS key indistinguishability security notions.** In Chapter 5 we illustrated an HKAS which is secure in the KR model but insecure in the S-KR model. It is an open problem to either find an HKAS which is secure in the KI model but insecure in the S-KI model, or to show that the KI and S-KI notions are (polynomially) equivalent.

Bibliography

[1] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In David Naccache, editor, *CT-RSA*, volume 2020 of *Lecture Notes in Computer Science*, pages 143–158. Springer, 2001.

[2] Michel Abdalla, Dario Fiore, and Vadim Lyubashevsky. From selective to full security: Semi-generic transformations in the standard model. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 316–333. Springer, 2012.

[3] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. The EM side-channel(s). In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 29–45. Springer, 2002.

[4] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In Reingold [116], pages 474–495.

[5] Selim G. Akl and Peter D. Taylor. Cryptographic solution to a problem of access control in a hierarchy. *ACM Trans. Comput. Syst.*, 1(3):239–248, 1983.

[6] Werner Alexi, Benny Chor, Oded Goldreich, and Claus-Peter Schnorr. RSA and Rabin functions: Certain parts are as hard as the whole. *SIAM J. Comput.*, 17(2):194–209, 1988.

[7] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In Halevi [83], pages 36–54.

[8] Mikhail J. Atallah, Keith B. Frikken, and Marina Blanton. Dynamic and efficient key management for access hierarchies. In Atluri et al. [12], pages 190–202.

[9] Mikhail J. Atallah, Marina Blanton, Nelly Fazio, and Keith B. Frikken. Dynamic and efficient key management for access hierarchies. *ACM Trans. Inf. Syst. Secur.*, 12(3), 2009.

[10] Giuseppe Ateniese, Alfredo De Santis, Anna Lisa Ferrara, and Barbara Masucci. Provably-secure time-bound hierarchical key assignment schemes. In Juels et al. [96], pages 288–297.

[11] Giuseppe Ateniese, Alfredo De Santis, Anna Lisa Ferrara, and Barbara Masucci. Provably-secure time-bound hierarchical key assignment schemes. *J. Cryptology*, 25(2):243–270, 2012.

[12] Vijay Atluri, Catherine Meadows, and Ari Juels, editors. *Proceedings of the 12th ACM Conference on Computer and Communications Security, CCS 2005, Alexandria, VA, USA, November 7-11, 2005*, 2005. ACM.

[13] Elaine Barker, Don Johnson, and Miles Smid. NIST special publication 800-56A: Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography (revised), March 2007.

[14] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford E. Tavares, editors, *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer, 2005.

[15] Mihir Bellare. Practice-oriented provable-security. In Eiji Okamoto, George I. Davida, and Masahiro Mambo, editors, *ISW*, volume 1396 of *Lecture Notes in Computer Science*, pages 221–231. Springer, 1997.

[16] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Juels et al. [96], pages 390–399.

[17] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer, 1993.

[18] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.

[19] Mihir Bellare and Bennet S. Yee. Forward-security in private-key cryptography. In Marc Joye, editor, *CT-RSA*, volume 2612 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2003.

[20] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Koblitz [98], pages 1–15.

[21] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Preneel [113], pages 139–155.

[22] Daniel J. Bernstein. Curve25519: New Diffie-Hellman speed records. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 207–228. Springer, 2006.

[23] Richard E. Blahut. *Algebraic Codes for Data Transmission*. Cambridge University Press, 2003.

[24] G. R. Blakley and David Chaum, editors. *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, 1985. Springer.

[25] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 1998.

[26] Lenore Blum, Manuel Blum, and Mike Shub. A simple unpredictable pseudo-random number generator. *SIAM J. Comput.*, 15(2):364–383, 1986.

[27] Manuel Blum and Shafi Goldwasser. An efficient probabilistic public-key encryption scheme which hides all partial information. In Blakley and Chaum [24], pages 289–302.

[28] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo random bits. In *FOCS*, pages 112–117. IEEE Computer Society, 1982.

[29] Carlo Blundo, Alfredo De Santis, Amir Herzberg, Shay Kutten, Ugo Vaccaro, and Moti Yung. Perfectly secure key distribution for dynamic conferences. *Inf. Comput.*, 146(1):1–23, 1998.

[30] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2003.

[31] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Franklin [61], pages 443–459.

[32] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[33] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *IACR Cryptology ePrint Archive*, 2002:80, 2002. Also appeared in volume 324 of *Journal of Contemporary Mathematics*, pages 71–90, 2002.

[34] Colin Boyd, Wenbo Mao, and Kenneth G. Paterson. Key agreement using statically keyed authenticators. In Markus Jakobsson, Moti Yung, and Jianying Zhou, editors, *ACNS*, volume 3089 of *Lecture Notes in Computer Science*, pages 248–262. Springer, 2004.

[35] Xavier Boyen, Qixiang Mei, and Brent Waters. Direct chosen ciphertext security from identity-based techniques. In Atluri et al. [12], pages 320–329.

[36] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145. IEEE Computer Society, 2001.

[37] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer, 2001.

[38] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In Jeffrey Scott Vitter, editor, *STOC*, pages 209–218. ACM, 1998.

[39] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[40] David Cash, Eike Kiltz, and Victor Shoup. The twin Diffie-Hellman problem and applications. In Nigel P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 127–145. Springer, 2008.

[41] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 523–552. Springer, 2010.

[42] Çağatay Çapar, Dennis Goeckel, Kenneth G. Paterson, Elizabeth A. Quaglia, Don Towsley, and Murtaza Zafer. Signal-flow-based analysis of wireless security protocols. *Inf. Comput.*, 226:37–56, 2013.

[43] Sanjit Chatterjee, Darrel Hankerson, and Alfred Menezes. On the efficiency and security of pairing-based protocols in the type 1 and type 4 settings. In M. Anwar Hasan and Tor Helleseth, editors, *WAIFI*, volume 6087 of *Lecture Notes in Computer Science*, pages 114–134. Springer, 2010.

[44] Liqun Chen, Zhaohui Cheng, and Nigel P. Smart. Identity-based key agreement protocols from pairings. *Int. J. Inf. Sec.*, 6(4):213–241, 2007.

[45] Tzer-Shyong Chen and Yu-Fang Chung. Hierarchical access control based on chinese remainder theorem and symmetric algorithm. *Computers & Security*, 21(6):565–570, 2002.

[46] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *IACR Cryptology ePrint Archive*, 2001:108, 2001. Also appeared in volume 33 of *SIAM Journal on Computing*, pages 167–226, 2003.

[47] Jason Crampton, Keith M. Martin, and Peter R. Wild. On key assignment for hierarchical access control. In *CSFW*, pages 98–111. IEEE Computer Society, 2006.

[48] Jason Crampton, Rosli Daud, and Keith M. Martin. Constructing key assignment schemes from chain partitions. In Sara Foresti and Sushil Jajodia,

editors, *DBSec*, volume 6166 of *Lecture Notes in Computer Science*, pages 130–145. Springer, 2010.

[49] Ivan Damgård. Collision free hash functions and public key signature schemes. In David Chaum and Wyn L. Price, editors, *EUROCRYPT*, volume 304 of *Lecture Notes in Computer Science*, pages 203–216. Springer, 1987.

[50] Paolo D'Arco, Alfredo De Santis, Anna Lisa Ferrara, and Barbara Masucci. Variations on a theme by Akl and Taylor: Security and tradeoffs. *Theor. Comput. Sci.*, 411(1):213–227, 2010.

[51] Alexander W. Dent. A note on game-hopping proofs. *IACR Cryptology ePrint Archive*, 2006:260, 2006.

[52] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, August 2008.

[53] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[54] Robert P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, 51(1):161–166, 1950.

[55] Yevgeniy Dodis, Rosario Gennaro, Johan Håstad, Hugo Krawczyk, and Tal Rabin. Randomness extraction and key derivation using the CBC, Cascade and HMAC modes. In Franklin [61], pages 494–510.

[56] Yevgeniy Dodis, Jonathan Katz, Adam Smith, and Shabsi Walfish. Composability and on-line deniability of authentication. In Reingold [116], pages 146–162.

[57] Régis Dupont and Andreas Enge. Provably secure non-interactive key distribution based on pairings. *Discrete Applied Mathematics*, 154(2):270–276, 2006.

[58] Sebastian Faust, Eike Kiltz, Krzysztof Pietrzak, and Guy N. Rothblum. Leakage-resilient signatures. In Daniele Micciancio, editor, *TCC*, volume 5978 of *Lecture Notes in Computer Science*, pages 343–360. Springer, 2010.

[59] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO*,

volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[60] Roger Fischlin and Claus-Peter Schnorr. Stronger security proofs for RSA and Rabin bits. *Journal of Cryptology*, 13(2):221–244, 2000.

[61] Matthew K. Franklin, editor. *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, 2004. Springer.

[62] Eduarda S. V. Freire and Kenneth G. Paterson. Provably secure key assignment schemes from factoring. In Udaya Parampalli and Philip Hawkes, editors, *ACISP*, volume 6812 of *Lecture Notes in Computer Science*, pages 292–309. Springer, 2011.

[63] Eduarda S. V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. Non-interactive key exchange. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 254–271. Springer, 2013.

[64] Eduarda S. V. Freire, Dennis Hofheinz, Kenneth G. Paterson, and Christoph Striecks. Programmable hash functions in the multilinear setting. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 513–530. Springer, 2013.

[65] Eduarda S. V. Freire, Kenneth G. Paterson, and Bertram Poettering. Simple, efficient and strongly KI-secure hierarchical key assignment schemes. In Ed Dawson, editor, *CT-RSA*, volume 7779 of *Lecture Notes in Computer Science*, pages 101–114. Springer, 2013.

[66] Gerhard Frey, Michael Müller, and Hans-Georg Rück. The Tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. *IEEE Transactions on Information Theory*, 45(5):1717–1719, 1999.

[67] S.D. Galbraith. *Advances in Elliptic Curve Cryptography*, chapter 9: Pairings, pages 183–213. Cambridge University Press, LMS 317, Cambridge, 2005.

[68] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[69] David Galindo. Boneh-Franklin identity based encryption revisited. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 791–802. Springer, 2005.

[70] David Galindo, Rodrigo Roman, and Javier Lopez. A killer application for pairings: Authenticated key establishment in underwater wireless sensor networks. In *CANS*, pages 120–132, 2008.

[71] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In Blakley and Chaum [24], pages 10–18.

[72] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2013. Full version available at `http://eprint.iacr.org/2012/610`.

[73] Rosario Gennaro, Shai Halevi, Hugo Krawczyk, Tal Rabin, Steffen Reidt, and Stephen D. Wolthusen. Strongly-resilient and non-interactive hierarchical key-agreement in MANETs. In Sushil Jajodia and Javier López, editors, *ESORICS*, volume 5283 of *Lecture Notes in Computer Science*, pages 49–65. Springer, 2008.

[74] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In Johnson [92], pages 25–32.

[75] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *FOCS*, pages 464–479. IEEE Computer Society, 1984.

[76] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[77] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *STOC*, pages 365–377. ACM, 1982.

[78] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[79] 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.

[80] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[81] Hua Guo, Yi Mu, Zhoujun Li, and Xiyong Zhang. An efficient and non-interactive hierarchical key agreement protocol. *Computers & Security*, 30(1): 28–34, 2011.

[82] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember: Cold boot attacks on encryption keys. In Paul C. van Oorschot, editor, *USENIX Security Symposium*, pages 45–60. USENIX Association, 2008.

[83] Shai Halevi, editor. *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, volume 5677 of *Lecture Notes in Computer Science*, 2009. Springer.

[84] Goichiro Hanaoka, Takahiro Matsuda, and Jacob C. N. Schuldt. On the impossibility of constructing efficient key encapsulation and programmable hash functions in prime order groups. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 812–831. Springer, 2012.

[85] Lein Harn and Hung-Yu Lin. A cryptographic key generation scheme for multilevel data security. *Computers & Security*, 9(6):539–546, 1990.

[86] Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 21–38. Springer, 2008.

[87] Dennis Hofheinz and Eike Kiltz. The group of signed quadratic residues and applications. In Halevi [83], pages 637–653.

[88] Dennis Hofheinz and Eike Kiltz. Practical chosen ciphertext secure encryption from factoring. In Antoine Joux, editor, *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 313–332. Springer, 2009.

[89] Dennis Hofheinz, Tibor Jager, and Eike Kiltz. Short signatures from weaker assumptions. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 647–666. Springer, 2011.

[90] Dennis Hofheinz, Eike Kiltz, and Victor Shoup. Practical chosen ciphertext secure encryption from factoring. *J. Cryptology*, 26(1):102–118, 2013.

[91] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In Ueli M. Maurer, editor, *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 143–154. Springer, 1996.

[92] David S. Johnson, editor. *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, 1989. ACM.

[93] Antoine Joux. A one round protocol for tripartite Diffie-Hellman. In Wieb Bosma, editor, *ANTS*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer, 2000.

[94] Antoine Joux. A one round protocol for tripartite Diffie-Hellman. *J. Cryptology*, 17(4):263–276, 2004.

[95] Antoine Joux and Kim Nguyen. Separating decision Diffie-Hellman from computational Diffie-Hellman in cryptographic groups. *J. Cryptology*, 16(4):239–247, 2003.

[96] Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors. *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*, 2006. ACM.

[97] Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 703–720. Springer, 2009.

[98] Neal Koblitz, editor. *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, 1996. Springer.

[99] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Koblitz [98], pages 104–113.

[100] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[101] Hugo Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 546–566. Springer, 2005.

[102] Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *NDSS*. The Internet Society, 2000.

[103] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 465–485. Springer, 2006.

[104] Anna Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 597–612. Springer, 2002.

[105] Stephen J. MacKinnon, Peter D. Taylor, Henk Meijer, and Selim G. Akl. An optimal algorithm for assigning cryptographic keys to control access in a hierarchy. *IEEE Trans. Computers*, 34(9):797–802, 1985.

[106] Kevin S. McCurley. A key distribution system equivalent to factoring. *J. Cryptology*, 1(2):95–105, 1988.

[107] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Florida, 1997.

[108] Alfred Menezes, Tatsuaki Okamoto, and Scott A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993.

[109] Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *CRYPTO*, volume 293 of *Lecture Notes in Computer Science*, pages 369–378. Springer, 1987.

[110] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In Moni Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, 2004.

[111] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In Halevi [83], pages 18–35.

[112] Kenneth G. Paterson and Sriramkrishnan Srinivasan. On the relations between non-interactive key distribution, identity-based encryption and trapdoor discrete log groups. *Des. Codes Cryptography*, 52(2):219–241, 2009.

[113] Bart Preneel, editor. *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*, 2000. Springer.

[114] M. O. Rabin. Digital signatures and public key functions as intractable as factorization. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1979.

[115] M. Ramkumar, N. Memon, and R. Simha. A hierarchical key pre-distribution scheme. In *Electro Information Technology, 2005 IEEE International Conference*, May 2005. doi: 10.1109/EIT.2005.1626994.

[116] Omer Reingold, editor. *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, volume 5444 of *Lecture Notes in Computer Science*, 2009. Springer.

[117] E. Rescorla. Diffie-Hellman key agreement method. RFC 2631 (Proposed Standard), June 1999. URL `http://www.ietf.org/rfc/rfc2631.txt`.

[118] Thomas Ristenpart and Scott Yilek. The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks. In Moni Naor, editor, *EUROCRYPT*, volume 4515 of *Lecture Notes in Computer Science*, pages 228–245. Springer, 2007.

[119] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[120] Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In Bimal K. Roy and Willi Meier, editors, *FSE*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2004.

[121] Ron Rothblum. On the circular security of bit-encryption. In Amit Sahai, editor, *TCC*, volume 7785 of *Lecture Notes in Computer Science*, pages 579–598. Springer, 2013.

[122] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *Proceedings of the 2000 Symposium on Cryptography and Information Security (SCIS 2000)*, January 2000.

[123] Alfredo De Santis, Anna Lisa Ferrara, and Barbara Masucci. Efficient provably-secure hierarchical key assignment schemes. In Ludek Kucera and Antonín Kucera, editors, *MFCS*, volume 4708 of *Lecture Notes in Computer Science*, pages 371–382. Springer, 2007.

[124] Laurent Sauvage, Sylvain Guilley, and Yves Mathieu. Electromagnetic radiations of FPGAs: High spatial resolution cartography and attack on a cryptographic module. *TRETS*, 2(1), 2009.

[125] Hovav Shacham. *New Paradigms in Signature Schemes*. PhD thesis, Stanford University, December 2005.

[126] Claude E. Shannon. Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4):656–715, October 1949.

[127] Victor R. L. Shen and Tzer-Shyong Chen. A novel key management scheme based on discrete logarithms and polynomial interpolations. *Computers & Security*, 21(2):164–171, 2002.

[128] Z. Shmuely. *Composite Diffie-Hellman public-key generating systems are hard to break*. Department of Computer Science: Technical report. Department of

Computer Science, TECHNION, 1985. http://books.google.co.uk/books?id=YTUTRAAACAAJ.

[129] Victor Shoup. On formal models for secure key exchange (version 4), November 1999. http://shoup.net/papers/.

[130] Victor Shoup. Using hash functions as a hedge against chosen ciphertext attack. In Preneel [113], pages 275–288.

[131] Victor Shoup. OAEP reconsidered. *J. Cryptology*, 15(4):223–249, 2002.

[132] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, 2004:332, 2004.

[133] Victor Shoup. *A computational introduction to number theory and algebra.* Cambridge University Press, 2006.

[134] Wen-Guey Tzeng. A secure system for data access based on anonymous authentication and time-dependent hierarchical keys. In Ferng-Ching Lin, Der-Tsai Lee, Bao-Shuh Paul Lin, Shiuhpyng Shieh, and Sushil Jajodia, editors, *ASIACCS*, pages 223–230. ACM, 2006.

[135] Serge Vaudenay. Security flaws induced by CBC padding - applications to SSL, IPSEC, WTLS ... In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 534–546. Springer, 2002.

[136] Umesh V. Vazirani and Vijay V. Vazirani. Efficient and secure pseudo-random number generation. In Blakley and Chaum [24], pages 193–202.

[137] Shyh-Yih Wang and Chi-Sung Laih. Merging: An efficient solution for a time-bound hierarchical key assignment scheme. *IEEE Trans. Dependable Sec. Comput.*, 3(1):91–100, 2006.

[138] Brent Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, 2005.

[139] André Weil. Sur les fonctions algébriques à corps de constantes fini. *C. R. Acad. Sci. Paris 210*, pages 592–594, 1940.

[140] Hugh C. Williams. A modification of the RSA public-key encryption procedure. *IEEE Transactions on Information Theory*, 26(6):726–729, 1980.

[141] Tzong-Chen Wu and Chin-Chen Chang. Cryptographic key assignment scheme for hierarchical access control. *Comput. Syst. Sci. Eng.*, 16(1):25–28, 2001.

[142] Jyh-Haw Yeh, Randy Chow, and Richard Newman. A key assignment for enforcing access control policy exceptions. In *Int. Symposium on Internet Technology*, pages 54–59, 1998.